DATORBELASTNING

MATS JONSSON

RE-166 Oktober 1975
Inst. för Reglerteknik
Lunds Tekniska Högskola

D A T O R B E L A S T N I N G

EXAMENSARBETE

vid

ASEA, VÄSTERÅS

för

Institutionen för REGLERTEKNIK

LUNDS TEKNISKA HÖGSKOLA

av

MATS JONSSON

1975

INNEHÅLLSFÖRTECKNING

K A P I T E L 1

SAMMANFATTNING

1. Sammanfattning

Då realtidssystem tenderar bli alltmer komplexa med mer svåröverskådliga arbetsuppgifter, ökar behovet att objektivt mäta belastningen på systemet.

Detta examensarbete försöker ge ett antal olika definitionsgrunder för begreppet belastning. Därefter beskrives ett antal lösningsförslag för mätning enligt de olika definitionerna.

Slutligen redovisas ett komplett "programpaket" för mätning enligt ett av dessa lösningsförslag. Paketet som är anpassat till MODCOMP, MAX III mäter CPU - belastningen utan åtskillnad av olika funktioner. Programmen är skrivna i FORTRAN och hur anpassning till andra datorer än MODCOMP skall ske beskrives även.

Summary

As realtimesystems has a tendency to grow more and more complexed with jobs that are harder to control, the need to measure the load of the computer increase.

This work tries to give some different grounds for a definition of "computerload" and thereafter describes suggestions to measure the load based on thease definitions.

Finally, a complete set of programs for measuring according to one of theese suggestions is presented. The set of programs is fitted to MODCOMP, MAX III and measures the total CPU - load without separation of different funktions. The programs are written in FORTRAN - language and howe to coordinate them whith other systems than MODCOMP is also described.

K A P I T E L 2

INLEDNING

2. Inledning

Realtidssystem tenderar att bli alltmer komplicerade och komplexà till sin uppbyggnad. Mjukvaran utformas för att kunna styra och bearbeta större och mer avancerade funktioner. Antalet yttre enheter ökar och möjligheten till en överblick av hela systemet minskar i takt med detta. Olika delar belastas enligt komplicerade mönster som är svåra att förutsäga. Behovet av att på något sätt objektivt mäta belastningen framstår därför som alltmer nödvändig.

Detta examensarbete försöker definiera begreppet belastning, vad som kan utgöra belastning och hur detta kan mätas. Omfattningen för arbetet var ifrån början sammanfattade till följande tre punkter:

1. Definiera begreppet belastning av centralenheten hos ett realtidssystem.
2. Teoretiskt redogöra för hur belastningen kan mätas.
3. Införa i en realtidsmonitor en rutin som möjliggör avläsning av de olika programmens belastning på centralenheten.

Ganska snart framkom det emellertid att punkt 3 utgorde en stor begränsning. Att inrikta sig på en mätrutin i monitorn för mätning av enskilda programs belastning utesluter flera möjliga mätmetoder. Därför kan punkt 3 sägas ha omformats till:

3*. Till ett system foga mjuk eller hårdvara för mätning av belastningen definierad i punkt 2.

Kapitel 3 behandlar olika principer för mätning. Dels uppföljning av alla ändringar i systemet och dels stickprovstagning av detsamma. Möjliga sätt att behandla och

presentera resultatet genomgås. Slutligen ges ett antal möjliga definitionsgrunder med motivering varför mätning efter just dessa. För och nackdelar samt utdatas form anges också.

Lösningsförslag grupperade efter de olika definitionsgrunderna skisseras i kapitel 4. Varje förslag beskrives kortfattat med angivande av positiva och negativa egenskaper.

I kapitel 5 ges några synpunkter på vad som behöver utföras vid en mätning i praktiken.

Slutligen kapitel 6, där har ett komplett "mätpaket" utvecklats efter ett av lösningsförslagen i kapitel 4. Paketet mäter CPU - belastningen utan särskillnad av olika funktioner utförda av operativsystem respektive användarprogram och är anpassat till MODCOMP, MAX III. Endast mjukvara användes och består av fem stycken program skrivna i FORTRAN. Programmen beskrives genom flödesschema och listor över instruktionerna. Vidare anges hur anpassning till ett befintligt system skall ske och vilka ändringar som måste utföras för användning på annan dator än MODCOMP. Några olika simulerade belastningsfall visas med kommentarer om olika möjligheter som variation av vissa mätparametrar ger.

En belastningsmätning enligt de metoder som skisseras i de följande kan endast användas för mätning på redan uppkopplat system. Vid utveckling av nya system är det nödvändigt att veta den ungefärliga utnyttjandegraden som olika enheter kommer att utsättas för. Detta för att redan på ett tidigt stadium kunna dimensionera program och maskinvara.

Därför bör alltid, vid större system, en simulering ske av den arbetssituation som datorn skall arbeta i. Simuleringen skall alltså ske innan systemet utvecklats helt och en eventuell belastningsmätning kan ske för att verifiera simuleringsresultatet.

För övrigt bör en belastningsmätning vara av intresse när ett system på något sätt ändrats eller utökats. Även regelbundna kontroller med jämna mellanrum, som ett led i underhållet, kan vara befogade.

Västerås 75-10-02

Mats Jonsson

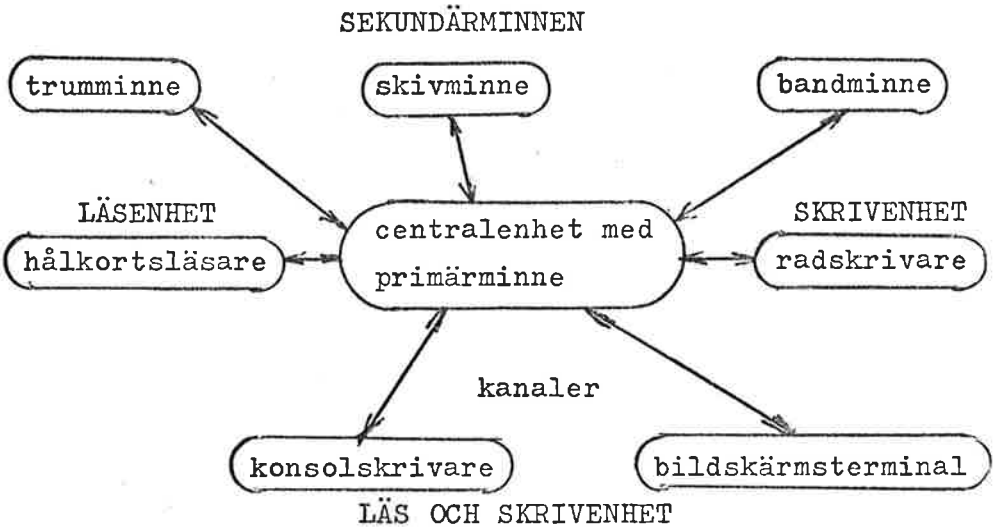
K A P I T E L 3

DEFINITION AV BEGREPPET BELASTNING

3. Definition av begreppet belastning.

3.0 Allmänt

Det finns ett flertal möjligheter att definiera begreppet belastning vad det gäller datorer. Eftersom ett komplett datorsystem normalt innehåller många olika enheter (centralenhet med primärminne, läs och skrivutrustning, olika typer av sekundärminnen m.m.) kan en belastningsmätning vara befogad på en eller flera av dessa enheter. Se figur 3.1



Figur 3.1 Centralenhetens kommunikation med I/O-enheter via kanaler.

Var och i vilken omfattning mätning bör ske i ett visst system beror på flera saker. Beroende på systemets fysiska dimensionering och dess användningsområde kan flaskhalsar uppträda på flera ställen. Här kommer endast belastning på centralenhet med primärminne och I/O-kanaler att behandlas. Detta på grund av att dessa delar av systemet bedömts som mest intressanta. I centralenheten sker "tankearbetet" och resultatet av detta distribueras via I/O-kanaler till enheter som presenterar detta för användaren. Således täcker en mätning av belastningen på dessa delar en stor del av det viktigaste som sker i systemet. När mätning sker på centralenhet med primärminne kan detta göras på ett flertal sätt. Mer om detta senare.

En uppdelning kan göras efter sättet att mäta. Den tid som en viss resurs utnyttjas, eller det antal gånger som samma resurs användes är två möjliga indelningsgrunder. Om inte hela resursen tages i anspråk samtidigt kan det vara intressant att veta hur stor del.

Utgående ifrån vad som sagt ovan har följande indelning gjorts. Nummerbeteckningen visar i vilket avsnitt respektive rubrik behandlas.

3.1 Exekveringstid.
3.1.1 Total exekveringstid för operativsystem plus applikationsprogram.
3.1.2 Exekveringstid för operativsystem respektive applikationsprogram.
3.1.3 Exekveringstid för operativsystem respektive varle applikationsprogram var för sig.
3.1.4 Exekveringtid för varje prioritetsnivå.

3.2 Exekveringsfrekvensen för varje applikationsprogram

3.3 Primärminnesbeläggning.

3.4 Kölängd.
3.4.1 Längden på kön i väntan på exekvering.
3.4.2 Längden på kön i väntan på I/O - enhet.
3.4.3 Längden på kön i väntan på primärminnesutrymme.

3.5 Utnyttjandegraden av I/O - kanaler.

Eftersom en eventuell överbelastnings orsak ofta är maskerad kan det vara önskvärt att kombinera flera olika mätmetoder för att få en så fullständig bild som möjligt av belastningssituationen. De uppställda definitionsgrunderna ger nämligen var för sig endast begränsad information. En sammanställning av möjliga utdata finns i nästa avsnitt.

3.0.1 Utdatas typ och form

Typen av utdata som kan erhållas ifrån de i förra avsnittet presenterade definitionsgrunderna varierar och en sammanställning finns i figur 3.2 nedan.

DEFINITIONSGRUND

Total exekveringstid för operativsystem och applikationsprogram.
: Exekveringstid för op.system resp. applikationsprogram.
: : Exekveringstid för op.system resp. varje appl.program var för sig.
: : : Exekveringstid för varje prioritetsnivå.
: : : : Exekveringsfrekvens för varje applikationsprogram.
: : : : : Primärminnesbeläggning.
: : : : : : Längden på kön i väntan på exekvering.
: : : : : : : Längden på kön i väntan på I/O - enhet.
: : : : : : : : Utnyttjandegraden av I/O - kanaler.

	MÖJLIGA UTDATA
X X X X *	Total utnyttjandegrad av CPU.
X X X	Operativsystemets utnyttjandegrad av CPU.
X X X *	Samtliga applikationsprograms utnyttjandegrad av CPU.
X *	Varje applikationsprograms utnyttjandegrad av CPU.
* X *	Olika prioritetsnivåers utnyttjandegrad av CPU.
* X	Exekveringsfrekvensen för varje applikationsprogam.
X	Primärminnesbeläggning.
* *	Responstid, väntetid.
* *	Utnyttjandegrad av I/O - kanaler.

Figur 3.2 Tabell visande möjliga utdata vid mätning enligt de olika definitionsgrunderna. X - markering innebär att denna typ av utdata direkt är tillgänglig, medan * - markering säger att utdata inte direkt är tillgänglig men kan beräknas eller uppskattas. Hur detta kan ske i varje enskilt fall beskrives under respektive lösningsförslag i kapitel 4 samt i den följande defiitionsbeskrivningen.

En viktig fråga förutom typen av mätdata är i vilken enhet dessa bör anges. Vid mätning av utnyttjandegraden presenteras belastningen lämpligen som en procentsiffra. D.v.s. hur stor del av totala mättiden enheten varit utnyttjad.

Då exekveringsfrekvensen räknas bör resultatet redovisas som antalet exekveringar per tidsenhet, alltså frekvensen direkt. I fallet med primärminnesbeläggning är det lämpligt att ange procentuell beläggning som utstorhet. Det torde knappast vara av intresse att veta specifikt vilka minnesceller som är upptagna. Vid kölängdsmätning är antalet köande kanske inte alltid den intressantaste informationen utan endast om kö existerar eller inte.

Inte bara den enhet som resultatet presenteras i, kan variera, utan även dess form med avseende på kontinuerlig registrering, medelvärdesbildning o.s.v. Idet följande kommer ett antal möjligheter att behandlas med angivande av lämplighet vid olika typer av belastningsmätningar.

A. Beräkning av medelvärde

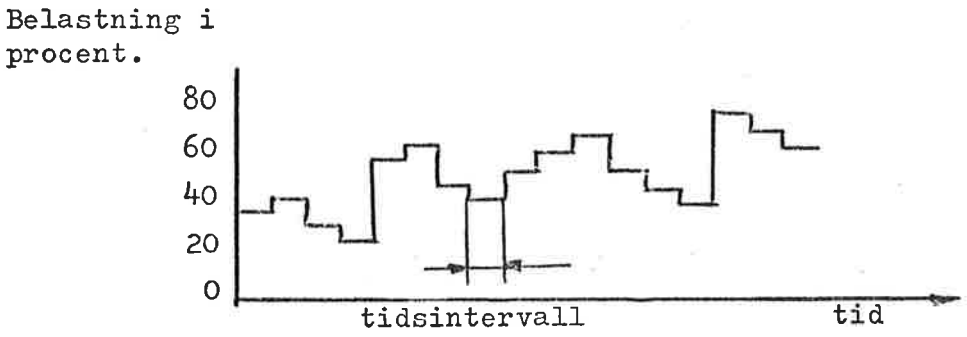
Den enklaste formen av resultatpresentation är att ange ett medelvärde för belastningen under längre tid. Minnesbehovet för lagring av delresultat blir mycket litet då summering av mätvärdena kan ske kontinuerligt under hela mättiden. Vid samtliga presenterade definitionsgrunder bör en medelvärdesbildning av utdata ske även om mera raffinierade metoder kommer till användning. För uppskattning av noggrannheten hos det beräknade medelvärdet, se vidare i avsnitt 3.0.2 där intervallskattning behandlas.

B. Resultatet presenterat tidsenhet för tidsenhet.

En tidskontinuerlig registrering och uppsamling av mätvärden är visserligen enkel att praktiskt utföra ,men kräver stort minnesutrymme för lagring av alla uppgifter. Dessutom ger det omfattande mätförfarandet stor egenbelastning på systemet och därav systematiska mätfel. Dess praktiska nytta torde inte heller vara speciellt stor såvida man inte i detalj önskar följa ett händelseförlopp.

Ett sådant fall kan i en processdator exempelvis vara när ett larm av något slag kommer. Larmet startar upp ett antal högt prioriterade program som belägger CPU, nödvändiga kanaler och yttre enheter så att andra program tvingas vänta. Sker en kontinuerlig presentation av belastningssituationen är det möjligt att avgöra när olika enheter "hämtat" sig från den extrema situationen och börjar fungera normalt igen.

En variant av kontinuerlig registrering som inte kräver lika stort minnesbehov och inte ger lika stor egenbelastning är någon form av "kvasikontinuerlig" registrering. Tekniken innebär att medelvärdesbilda under förhållandevis korta tidsintervall, hur korta beror på hur snabbt förändringarna sker i systemet. Medelvärdena under dessa intervall presenteras sedan enligt figur 3.3. Lämpligt val av tidsintervall kommenteras i kapitel 5, avsnitt 5.4.



Figur 3.3 Illustration av kvasikontinuerlig registrering.

Metoderna med kontinuerlig och kvasikontinuerlig mätning är inte speciellt anpassat till någon av belastningsgrunderna utan kan användas till samtliga.

C. Angivande av max och minvärden.

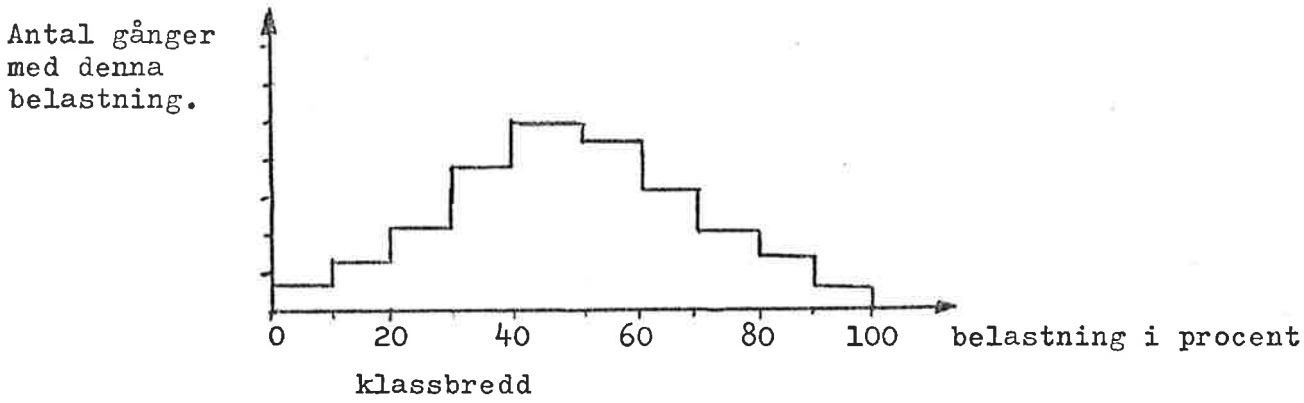
Att ange max och minvärdena vid presentationen blir mycket enkelt. Varje nytt mätvärde behöver endast kontrolleras emot föregående största och minsta värde för att avgöra om det överskrider tidigare gränser. Risken är emellertid att processen på vilken mätningen sker åtminstone någon gång under mättiden uppnår noll respektive 100% belastning. I så fall blir resultatet mindre intressant. Sannolikheten för att extremvärdena skall uppnås kan emellertid varieras gomom att variera tidsavståndet mellan resultatavläsningarna. Detta kommenteras vidare i avsnitt 5.4.

Vid mätning av utnyttjandegraden eller minnesbeläggningen är sannolikheten för noll och 100% belastning så stor att denna typ av resultatpresentation knappast ger någon nyttig information. Det är i första hand vid kölängdsmätning som resultatet kan vara intressant, speciellt då maxvärdet av kölängden kompletterat med angivande av medelvärde. Den största bristen är att ingenting säges om hur ofta maxvärdet uppnåtts eller nästan uppnåtts. För att få information om detta måste belastningens fördelning beräknas. Hur detta sker behandlas i nästa avsnitt.

D. Belastningens fördelning anges.

När belastningens fördelning skall anges delas möjliga belastningsvärden in i ett antal klasser. Belastningen mätes under kortare tidsintervall med medelvärdesbildning

och resultatet sorteras in i de olika klasserna och presenteras enligt figur 3.4. Beträffande val av lämpligt tidsintervall och därmed följande problem se avsnitt 5.4.



Figur 3.4 Exempel där belastningens fördelning angetts.

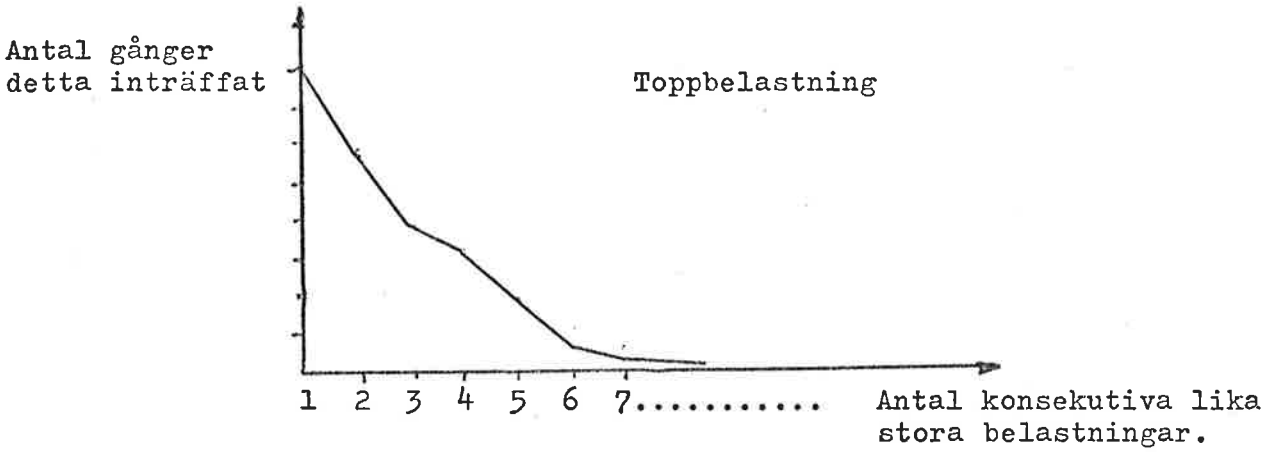
I figuren ovan är belastningen angiven i procent. Detta betyder emellertid inte att metoden inte är tillämpbar i fall med utdata angivet i annan enhet. Vid mätning av kölängder fungerar den utmärkt.

En angivelse av medelvärdet av belastningen bör vid denna metod alltid komplettera resultatet.

E. Antalet konsekutiva lika stora belastningar anges.

Vid denna form av resultatpresentation noteras det antal gånger i rad som belastningen haft ett visst värde. I första hand vid toppbelastning kan detta vara av intresse, för att kunna registrera skurar av maximalt utnyttjande av enheten på vilken mätning sker. Ett exempel visas i figur 3.5 på nästa sida.

Motiveringen för denna form av utdata är att belastningen ofta kommer väldigt ojämt med tillfälliga toppar då och då.



Figur 3.5 Illustration av metoden

Det är i första hand vid mätning av total utnyttjandegrad av centralenheten som ett presentationssätt av denna typ är intressant. Metoden bör när den användes alltid vara ett komplement till registrering av belastningsfördelningen enligt föregående avsnitt.

En sammanställning av de i avsnitt A - E beskrivna sätten att presentera resultatet och deras lämplighet vid mätning enligt olika utgångspunkter vid belastningsdefinitionen finn i figur 3.6 på nästa sida.

DEFINITIONSGRUND		PRESENTATIONSSÄTT				
		A	B	C	D	E
3.1.1 Total exekveringstid för op.system + appl.program.		4	2	1	3	3
3.1.2 Ex.tid för op.system respektive appl.program.		4	1	1	1	0
3.1.3 Ex.tid för op.system resp. appl.program var för sig.		4	1	1	1	0
3.1.4 Ex.tid för varje prioritetsnivå.		4	1	1	1	0
3.2 Ex.frekvensen för varje appl.program.		4	0	0	0	0
3.3 Primärminnesbeläggning.		4	1	1	2	1
3.4.1 Längden på kön i väntan på exekvering.	*)	4	1	3	3	1
3.4.2 Längden på kön i väntan på viss I/O-enhet.	*)	4	1	3	3	1
3.5 Utnyttjandegrad av I/O-kanaler.		4	1	1	1	2

Figur 3.6 Sammanställning över lämplig resultatpresentation. Sifferbeteckningarna har följande betydelse:

0 = Ej rimligt presentationssätt.
1 = Rimligt men knappast nödvändigt.
2 = Rimligt och eventuellt önskvärt.
3 = Önskvärt.
4 = Bör alltid finnas med.

*) Både C och D är knappast nödvändiga samtidigt.

3.0.2 Olika sätt att mäta belastningen

Två grundläggande skillnader att mäta belastningen kan särskiljas oberoende av de uppställda definitionsgrunderna. Här benäms dessa direkt respektive indirekt mätning och beskrivning av vad som menas i de båda fallen följer nedan.

A. Direkt mätning

En direkt mätning innebär att mätresultatet är ett direkt mått på belastningen, med reservation för eventuella systematiska fel. Beroende på vad som skall mätas blir emellertid tolkningen något olika.

I fallet med utnyttjandegraden mätes den tid som en enhet är upptagen. Till detta behövs därför någon form av klocka. När exekveringsfrekvensen skall kontrolleras sker detta med hjälp av en räknare som räknar upp var gång exekvering sker. Vid utrymmeskontroll, som mätning av primärminnesbeläggningen, måste upptaget utrymme summeras tills hela den tillgängliga minnesarean genomsökts. Kölängdsmätning får tillgå så att samtliga köande program summeras vid varje mättillfälle.

Fördelen med dessa typer av mätningar gentemot den indirekta, som här definierats som stickprovstagningar med vissa intervall, är att resultatet visar den sanna belastning som rådde vid mättillfället. När stickprov tages kan teoretiskt aldrig exakt belastning erhållas. Alla resultat blir behäftade med en viss osäkerhet. Hur stor beror på mängden insamlade data. Dessutom ger indirekt mätning alltid någon form av medelvärde på grund av den nödvändiga statistiska behandlingen.

Nackdelen med direkta mätningar är att betydligt fler uppgifter måste insamlas. Uppgifter som inte alltid är nödvändiga för tillräckligt god uppskattning om belastningen. Den osäkerhet som en indirekt mätning ger behöver

inte vara av någon betydelse då man troligen, i de flesta fall, endast är intresserad av ungefärliga belastningsvärden. Den metod som bör väljas i varje enskilt fall beror helt på vad som skall mätas och hur möjligheterna att genomföra mätningarna är.

B. Indirekt mätning

Som redan nämnts i föregående avsnitt innebär indirekt mätning någon form av stickprovstagning. Här kommer endast att behandlas två fall med mätning av CPU-belastning där enskilda programs exekveringstid mätes med stickprov. För kommentarer om för och nackdelar hänvisas till föregående beskrivning av direkt mätning.

De två fall av mätning som behandlas benämnes:

A - Mätning under förutbestämd tid.

B - Mätning under variabel tid.

Mätningen som sådan skiljer sig inte alls utan den enda skillnaden finns i behandlingen av mätresultatet. Normalt bör om inte mättiden måste fixeras, metod B vara att föredra eftersom den alltid kan ge önskad noggrannhet i skattningen av belastningen. Nackdelen är att den kräver större beräkningsarbete än metod A.

De använda beteckningarna är hämtade ur litteraturreferens 2 .

MÄTNING UNDER FÖRUTBESTÄMD TID

I detta fall sker mätning under så lång tid som operatören anser vara möjligt eller nödvändigt. Vid resultatpresentationen anges sedan mätvärdenas signifikans.

Som sagts tidigare kommer här endast mätning av CPU-belastning att behandlas. De utdata som erhålles i detta

fall kan sägas vara av formen enligt figur 3.7.

Program	P(1)	P(2)	P(3)	P(i)	P(N)
Antal markeringar	k_1	k_2	k_3	k_i	k_N

Figur 3.7 Utdatas form.

Statistiskt kan dessa anses beskrivna av binomialfördelningen med det i:te programmet givet av fördelningen X_i sådan att $X_i \in \text{Bin}(\, p_i, \, n\,)$. Variabeln p_i anger relativ belastning och n totala antalet stickprov d.v.s.

$$n = \sum_{i=1}^{N} k_i$$

Om det antages att n är stort kan binomialfördelningen approximeras med normalfördelningen och följande samband kan anses gälla för intervallskattning av mätvärdena,

$$Ip_i = p_i^{*} \pm \lambda_{\frac{\alpha}{2}} \cdot d_i \qquad \text{där} \quad d_i = \sqrt{\frac{p_i^{*}(\,1 - p_i^{*}\,)}{n}} \qquad (3.1)$$

p_i^{*} är en skattning av relativa belastningen p_i för det i:te programmet så att $p_i^{*} = k_i / n$. Faktorn $\lambda_{\frac{\alpha}{2}}$ är en konstant som antar olika värden beroende på vilken signifikans som önskas på skattningen. Några olika värden anges i figur 3.8.

Det skattade intervallets längd kan varieras dels med signifikansnivån och dels med medelfelet d_i. Avgörande för valet av lämplig signifikansnivå beror på hur stora kraven är på att den verkliga belastningen ligger inom intervallets längd.

Sannolikheten $(1-\alpha)\cdot 100$ med vilken mätvärdena ligger inom angivet intervall.	$\lambda_{\frac{\alpha}{2}}$
99 %	2.58
95 %	1.96
90 %	1.64
80 %	1.28
75 %	1.15

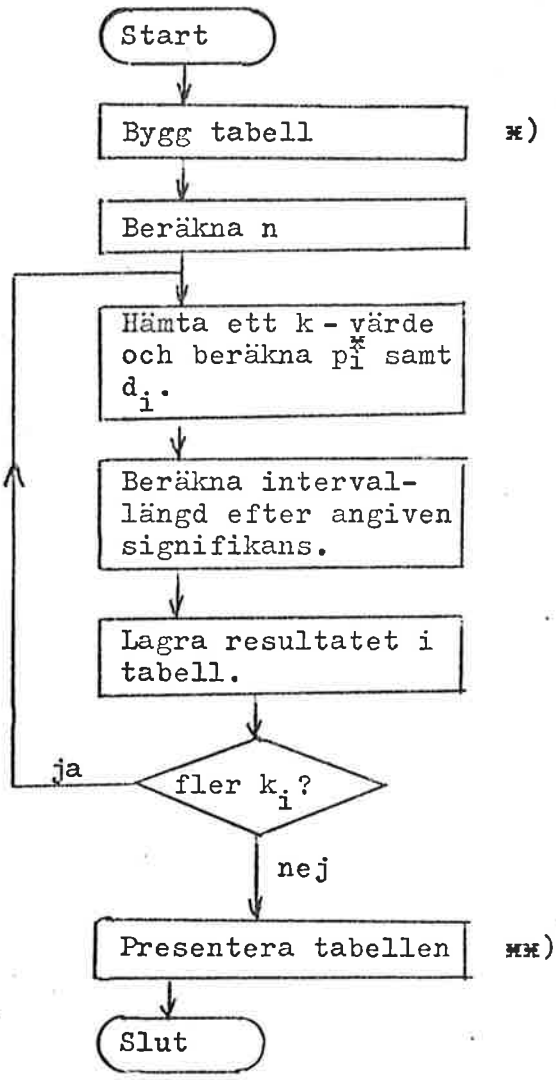
Figur 3.8 $\lambda_{\frac{\alpha}{2}}$ - värden för olika signifikansnivå.

Medelfelet är omvänt proportionellt emot roten ur antalet stickprov och detta medför att sambandet mellan mättid och intervalläng aproximativt blir:

$$\text{Intervallängd } Ip_i \approx \frac{1}{\sqrt{\text{mättid}}} \qquad (3.2)$$

För att halvera intervallet krävs alltså en fyrdubbling av mättiden.

Ett flödesschema över ett program som utför de nödvändiga beräkningarna i detta fall med förutbestämd mättid finns i figur 3.9 på nästa sida.



*) Denna tabell förutom programmets namn ha plats för p_i^* och $\lambda_{\frac{\alpha}{2}} \cdot d_i$.

**) Tabellens utseende kan i princip vara:

Program	Skattad relativ belastning	Skattningens noggrannhet
P(1)	p_1^*	$\pm \lambda_{\frac{\alpha}{2}} \cdot d_1$
P(2)	p_2^*	$\pm \lambda_{\frac{\alpha}{2}} \cdot d_2$
⋮	⋮	⋮

Figur 3.9 Flödesschema för nödvändiga beräkningar vid indirekt mätning under förutbestämd tid.

MÄTNING UNDER VARIABEL TID

Metodiken i detta fall liknar mycket den i föregående avsnitt beskrivna. De statistiska beräkningarna är av samma typ. Skillnaden är i första hand att, istället för mätning under en längre fastställd tid sker mätningen under kortare perioder med mellanliggande avbrott. I dessa avbrott kontrolleras om tillräcklig noggrannhet uppnåtts. Är så fallet avbrytes mätningen och resultatet presenteras, om inte fortsätter den ytterligare en period med ny efterföljand kontroll o.s.v.

Vid kontroll av tillräcklig noggrannhet är det intervallets längd som skall jämföras med före mätningen uppställda krav. Längden bör då ställas i relation till den skattade belastningens storlek så att fall enligt exemplet i figur 3.10 undvikes.

Antag att uppställda krav på intervallets längd är ± 0.5 %. Med två olika mätvärden blir resultatet:

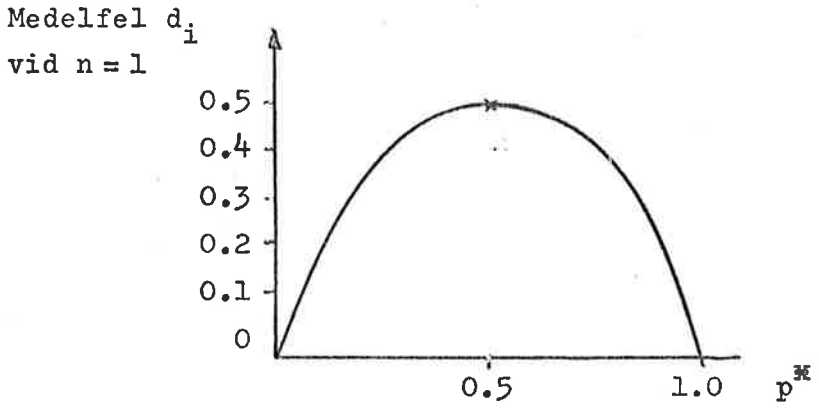
$$Ip_1 = 50\ \% \pm 0.5\ \%$$

$$Ip_2 = 0.1\ \% \pm 0.5\ \%$$

För program 1 blir kraven onödigt snäva, medan gränserna för program 2 blir orimligt vida.

Figur 3.10 Exempel på vad som kan hända vid felaktiga gränser på intervallängden.

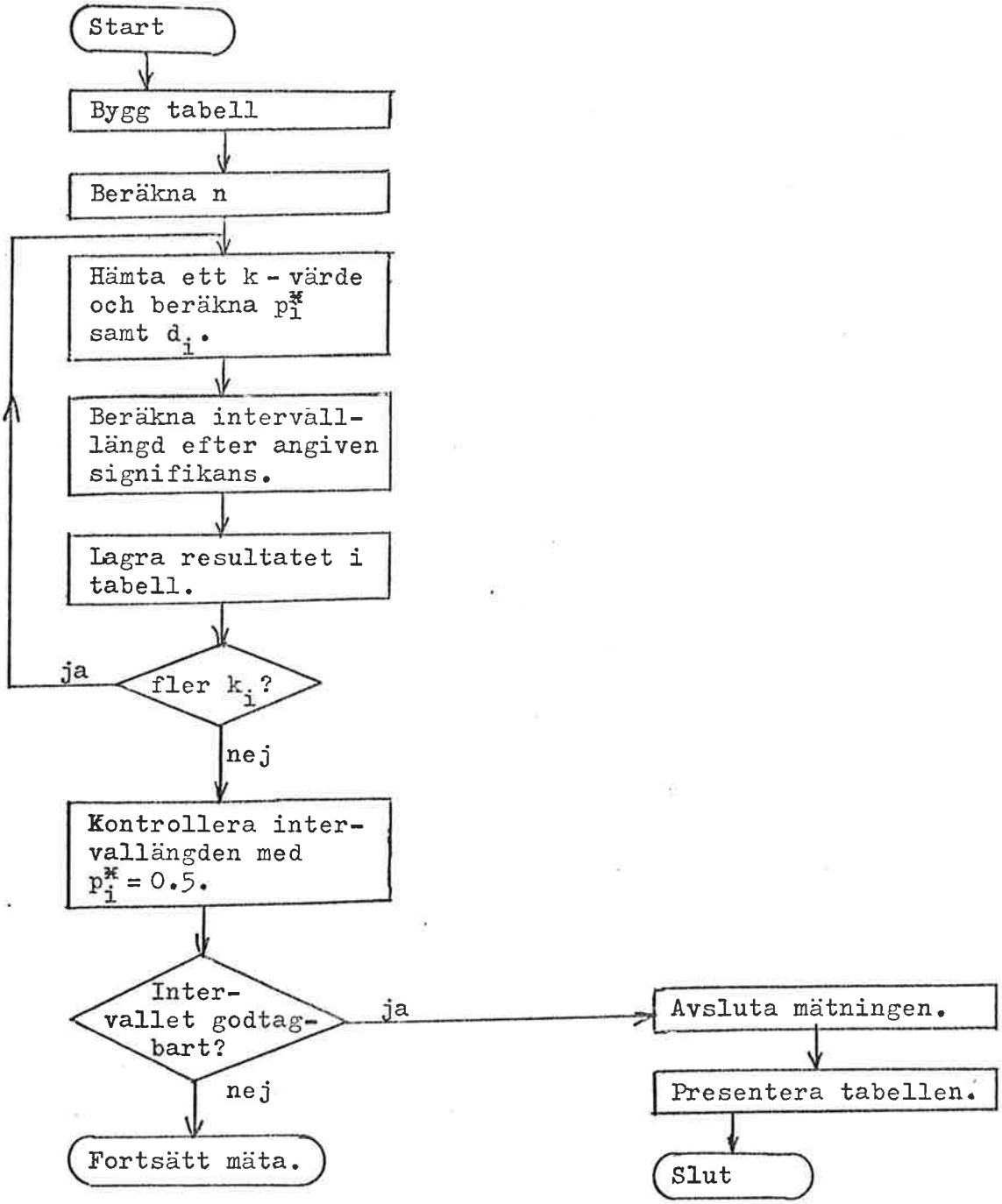
Eftersom intervallets längd beror på storleken av skattningen för belastningen och varierar enligt figur 3.11 bör sådana problem kunna undvikas genom att alltid kontrollera uppställda krav på intervallängden emot en skattning vid fixt p^{*}. Lämpligt p^{*} är 0.5 där medelfelet d_i har sitt maximum.



Figur 3.11 Medelfelets variation med storleken på p^*

I varje mätperiod adderas mätvärdena till de tidigare erhållna så att en succesivt bättre skattning alltid erhålles. Längden på perioden bör inte väljas alltför kort så att tiden för avbrottet blir stort i förhållande till själva mättiden. Lämpligt värde torde vara så stort att antal avbrott blir ungefär 4 - 6 stycken innan tillräcklig noggrannhet erhållits.

Ett flödesscema över lämplig utformning av det program som kallas in vid varje avbrott finns i figur 3.12. Jämför med figur 3.9.



Figur 3.12 Flödesschema för nödvändiga beräkningar vid indirekt mätning under variabel tid.

3.1 Exekveringstid

Vid definition av datorbelastning som exekveringstid har en nedbrytning gjorts i fyra nivåer beskrivna i de fyra avsnitten (3.1.1 - 3.1.4). Anledning till denna nedbrytning är att behovet för differentiering utav mätresultatet kan variera.

De tre första definitionerna är en direkt utveckling av varandra, som ger olika mängd information om belastningen, med 3.1.3 som den mest utvecklade. Detta på bekostnad av en betydligt mer komplicerad och tidskrävande behandling av mätdata.

Även det fjärde alternativet, 3.1.4 kan sägas vara en utveckling av de två första, men skiljer sig helt ifrån 3.1.3 i filosofi, eftersom mätningen inriktar sig på prioritetsnivåer och inte på enskilda program.

3.1.1 Total exekveringstid för operativsystem plus applikationsprogram.

Definition: Med total exekveringstid för operativsystem plus applikationsprogram menas den del av totalt förfluten tid som dessa tillsammans utnyttjar centralenheten för exekvering.

Motiv för användning: Ger upplysning om utnyttjandegraden av centralenheten, vilket är viktigt då man befarar att denna tenderar bli överbelastar. Även då CPU inte direkt kan anses vara överbelastad är det nyttigt att få veta dess utnyttjandegrad. Detta för att erhålla information om hur mycket ytterligare belastningen kan ökas utan omfattande mjuk eller hårdvaruingrepp.

Utdata: Utnyttjandegraden av CPU mätt med storheten tid.

Fördelar: - Lätt att praktiskt utföra då ingen information om enskilda program erfordras.

- Endast två storheter, aktiv och inaktiv tid behöver särskiljas.
- Ingrepp i operativsystemet behöver ej göras för att utföra mätningen.
- Om mätning sker av den tid då operativsystemet eller något applikationsprogram ej använder CPU kan, med kännedom om total mättid, utnyttjandegraden beräknas utan att mätningen stör dessa program. Mätförfarandet startar alltså upp endast då inget annat finns att göra.

Nackdelar:
- Ingen information om enskilda programs belastning erhålles utan omfattande extraarbete med programvis inkoppling.
- Då tidmätningen sker med någon form av intern tidsbas (cykeltid, klocktick o.s.v.) måste denna kalibreras om denna inte noggrant är definierad. Eventuella avvikelser ger nämligen direkt ett mätfel.

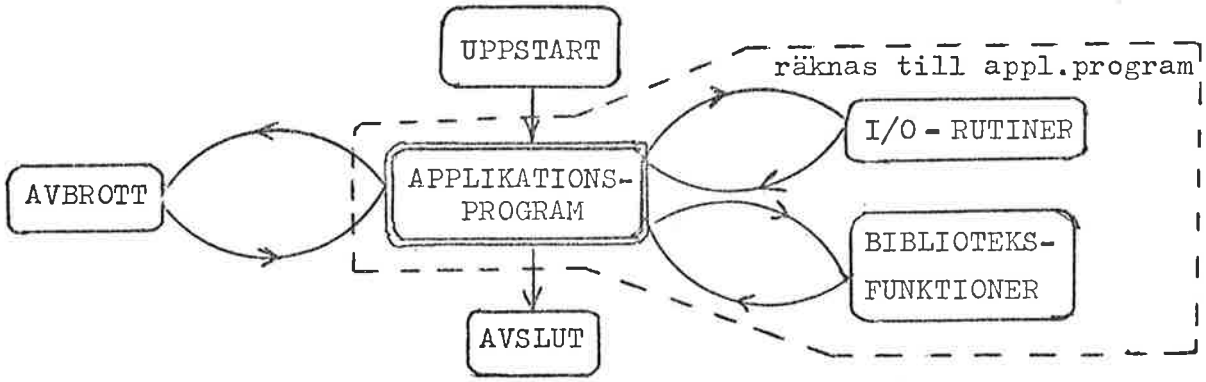
Lösningsförslag enligt denna definition återfinns i avsnitt 4.1.1.

3.1.2 Exekveringtid för operativsystem respektive applikationsprogram.

Definition: Med exekveringtid för operativsystem respektive applikationsprogram*) menas den del av totalt förfluten tid som dessa var för sig utnyttjar centralenheten för exekvering.

*) Eftersom applikationsprogammen utnyttjar operativsystemet för olika aktiviteter under exekveringens gång, se figur 3.13, kan konflikter uppstå vid uppdelningen av belastningen på de två fallen.

Till applikationsprogrammens belastning måste, för att entydighet skall uppnås, räknas alla rutiner som behövs för programmens exekvering utom uppstarts och avslutsrutiner.



Figur 3.13 Schematisk bild av vad som kan hända under exekvering av ett applikationsprogram.

Motiv för användning: Förutom de i 3.1.1 nämda motiven ger denna uppdelning av exekveringstiden i två delar ytterligare information. Kunskap om hur mycket operativsystemet utnyttjas kan visa om detta arbetar oproportioneligt stor del av tiden. Så kan exempelvis vara fallet när, på grund av bristfällig planering vid placeringen av program som primär eller sekundärminnesresidenta, onödig överföring tvingas ske mellan de olika minnena.

Utdata: Operativsystemets respektive samtliga applikationsprograms sammanlagda utnyttjandegrad av CPU mätt med storheten tid.

Fördelar:
- Endast en rutin som kontrollerar tiden för op.systemets respektive appl.progammens exekvering behöver införas.
- Ett fåtal mätstorheter att hålla reda på.
- Ingen information om enskilda program är nödvändig.
- Direkta uppgifter om operativsystemets belastning erhålles. Detta kan ge en uppfattning om dess effektivitet.

Nackdelar:
- Ingen information om enskilda programs exekveringstid erhålles såvida inte progammen aktiveras ett och ett med efterföljande mätning.
- Vissa ingrepp och ändringar i operativsystemet kan vara nödvändiga för att erhålla önskad information.

Lösningsförslag enligt denna definition återfinnes i avsnitt 4.1.2.

3.1.3 Exekveringtid för operativsystem respektive varje applikationsprogram var för sig.

Definition: Med exekveringstid för operativsystem respektive varje enskilt applikationsprogram*) menas den del av totalt förfluten tid som operativsystemet respektive varje enskilt applikationsprogram var för sig utnyttjar centralenheten för exekvering.

*) Samma kommentar angående applikationsprogrammens belastning som i föregående avsnitt.

Motiv för användning: Denna ytterligare nedbrytning av exekveringstiden motiveras av den direkta informationen om varje programs belastning som erhålles. Kännedom om varje programs belastning ger möjlighet att bättre planera indelningen i prioritetsnivåer. Dessutom gäller även de tidigare i 3.1.1 och 3.1.2 redovisade motiveringarna.

Utdata: Operativsystemets respektive varje enskilt applikationsprograms utnyttjandegrad av CPU mätt med storheten tid.

Fördelar:
- Varje enskilt programs belastning erhålles direkt.
- Med en eventuell särskillnad av operativsystemets olika delar, kan exempelvis I/O-rutinernas utnyttjandegrad mätas.

Nackdelar:
- Uppgift om alla i systemet befintliga program måste finnas.
- En tabell över samtliga program, för lagring av mätvärdena måste byggas innan själva mätningen kan starta.
- Vissa ingrepp i operativsystemet blir antagligen nödvändiga för att erhålla önskad information.
- Varje mätning tar relativt lång tid eftersom inte bara tiden skall noteras utan även program skall identifieras v. var gång. Detta ger hög egenbelastning.
- Användes den interna klockan för tidmätning kan eventuellt problem med klockupplösningen uppstå.

Lösningsförslag enligt denna definition återfinns i avsnitt 4.1.3

3.1.4 Exekveringstid för varje prioritetsnivå.

Definition: Med exekveringtiden för varje prioritetsnivå*) menas den del av totalt förfluten tid som varje prioritetsnivå utnyttjar centralenheten för exekvering.

*) Kommentaren om varje applikationsprograms belastning i avsnitt 3.1.2 gäller även här.

Operativsystemet kan i detta fall anses vara ett program på en egen unik prioritetsnivå.

Motiv för användning: Kunskap om fördelningen av exekveringstiden mellan olika prioritetsnivåer kan medverka till en effektivare fördelning av programmen mellan de olika nivåerna. Den totala utnyttjandgraden av centralenheten bör för högre prioritetsnivåer vara mindre än för lägre. En uppfattning om hur det i verkligheten förhåller sig i ett visst system erhålles säkrast genom en mätning på detta sätt. Att räkna antal program på varje nivå och uppskatta deras medelexekveringstid manuellt blir både tidsödande och osäkert.

Utdata: Samtliga program på en viss prioritetsnivås utnyttjande av CPU mätt med storheten tid.

Fördelar: - Om respektive program inom varje prioritetsnivå äktiveras var för sig ges samma möjlighet till bättre fördelning av programmen på de olika nivåerna som i 3.1.3. Detta med enklare mjukvara.

Nackdelar: - Kännedom om varje exekverande programs prioritet nödvändig.
- En tabell över samtliga prioritetsnivåer måste byggas innan mätningen kan startas. Tabellen användes för att lagra mätvärdena i .
- Troligen måste vissa ingrepp i operativsystemet göras.
- Mätproceduren tar ganska lång tid med kontroll av exekveringstider och prioritetsnivåer.

- Om den interna klockan användes som tidmätare kan problem med dess upplösning uppstå.

Lösningsförslag enligt denna definition återfinnes i avsnitt 4.1.4.

3.2 Exekveringsfrekvensen för varje applikationsprogram.

Definition: Med exekveringsfrekvensen för varje program menas det antal gånger per tidsenhet som ett visst program startas upp och avslutas.*)

*) Att programmet skall startas upp o c h avslutas innebär att registrering av exekveringen egentligen skall ske först e f t e r avslutning. Då emellertid normalt alla program som startas upp också avslutas blir felet försumbart om registreringen sker före eller under exekveringen.

Motiv för användning: Att mäta exekveringsfrekvensen torde vara enklare än att mäta exekveringstiden, då ingen klocka är nödvändig. Trots det kan approximativt samma information om programmens exekveringstid erhållas om varje programs medelexekveringstid är känd. Vidare kan det vid indelningen av program som primär respektive sekundärminnesresidenta vara värdefullt att veta hur ofta programmen användes. Detta för att program med hög utnyttjandefrekvens om möjligt bör läggas primärminnesresidenta för att undvika onödigt tidsspill vid inläsning ifrån yttre minne.

Utdata: Hur ofta varje program exekveras.

Fördelar:
- Inget behov av klocka. Därmed elimineras alla fel som kan tänkas bero på klockupplösning.
- Endast en räknare per program behövs för mätvärdesinsamling.
- Varje enskilt programs belastning erhålles.
- Mätstorheten, antal gånger, är entydig och lätt att tolka.

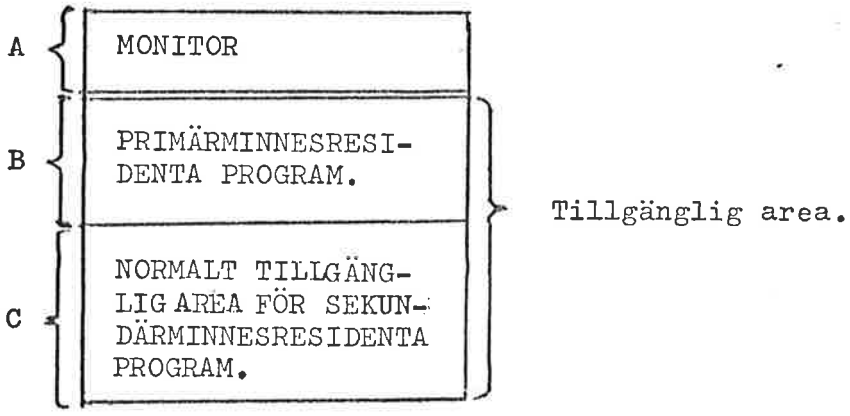
Nackdelar:- Kräver kännedom om varje enskilt program.
- En tabell över samtliga program måste byggas innan mätningen startar.
- Någon direkt upplysning om tidsbelastningen erhålles inte utan omfattande extraarbete med beräkning av exempelvis exekveringstider och tidsåtgång för I/O - överföringar.
- Ändringar i operativsystemet eller varje program troligen nödvändig.

lösningsförslag enligt denna definition återfinnes i avsnitt 4.2.

3.3 Primärminnesbeläggning.

Definition: Med primärminnesbeläggning menas den del av totalt tillgänglig minnesarea som är upptagen vid ett visst tillfälle.*)

*) Det är i första hand när minnet har en struktur enligt figur 3.14 som mätning av upptaget utrymme blir intressant. Vid annan uppbyggnad, exempelvis med partitioner, blir behovet av mätning betydligt mindre beroende på den annorlunda tekninken vid tilldelning av utrymme åt program.



Figur 3.14 Minnesdisposition i det fall då mätning kan anses befogad.

I denna framställning kommer en begränsning att ske till den typ av minnesdisposition som visas i figur 3.14 ovan.

Motiv för användning: Metodiken ger en uppfattning om hur väl minnet utnyttjas. Detta kan ligga till grund för en mer optimal indelning av program som primär eller sekundärminnesresidenta. Om exempelvis område C i figur 3.14 större delen av tiden har stort utrymme ledigt kan det vara befogat att öka område B, då på bekostnad av område C. En sådan ökning sker genom att antalet primärminnesresidenta program ökas och kan medföra färre överföringar till och från sekundärminnen. Följden för datoranvändaren blir att responstiden troligen minskar.

Utdata: Hur stor del av primärminnet som är upptaget vid ett visst tillfälle.

Fördelar:
- Eftersom mätstorheten är fysiskt utrymme blir mätdata entydiga och lättolkade.
- Uppgifter om alla programmen behöver inte finnas tillgängliga om kontroll av ledigt eller upptaget utrymme sker direkt i den minnesarea där uppgifter om minnesdisposionen lagras.

Nackdelar:
- Ger enbart information om primärminnet, vilket ger knapphändiga uppgifter för en utvärdering av belastningen i CPU med primärminne.
- Relativt komplicerat att få fram nödvändiga data.
- En ändring i operativsystemet är eventuellt nödvändig.
- En icke försumbar belastning på centralenheten blir troligen följden av det omfattande arbetet med framtagning av mätvärdena.

Lösningsförslag enligt denna definition återfinnes i avsnitt 4.3.

3.4 Kölängd

Vid definition av belastningen med hjälp av kölängden har en särskillnad av tre fall gjorts. De tre fallen som kommer att behandlas är:

3.4.1 Längden på kön i väntan på exekvering.
3.4.2 Längden på kön i väntan på I/O-enhet.
3.4.3 Längden på kön i väntan på primärminnesutrymme.

Eventuellt kan andra typer av kömätningar utföras men dessa tre har bedömts som de mest betydelsefulla. Kön i väntan på exekvering ger indikation på centralenhetens belastning medan kön i väntan på I/O-enhet talar om utnyttjandegraden på respektive enhet. Kön till primärminnet ger en uppfattning om dess beläggningsgrad.

En förenkling av metodiken är att i stället för räkning av antalet köande program endast kontrollera om kö finns eller inte. Detta behandlas närmare under lösningsförslagen i kapitel 4.4.

3.4.1 Längden på kön i väntan på exekvering

Definition: Med längden på kön i väntan på exekvering menas det antal program som väntar på att få påbörja*) sin exekvering.

*) Alltså inte program som avbrutits och väntar på att få fortsätta sin exekvering.

Motiv för användning: Längden på denna kö ger en uppfattning om belastningen på centralenheten, då en lång kö tyder på stor belastning. En konsekvens av detta är att även ett mått på responstiden erhålles. Vidare kan en lång kö tyda på att primärminnesutrymmet är hårt utnyttjat. Program tvingas vänta på inläsning eftersom utrymme saknas i minnet.

Utdata: Antal program som väntar på att få påbörja sin exekvering.

Fördelar:
- Enkla och entydiga mätvärden ger enkel efterbehandling av resultatet.
- Ingen kännedom om enskilda programs identitet är nödvändig.
- Kontrolleras endast om n å g o t progam är köande eller om kön är tom blir mätproceduren mycket enkel.

Nackdelar:
- Vissa manipulationer i operativsystemets arbetsareor är antagligen nödvändig för att få veta kölängden.
- Användes separata köer för varje prioritetsnivå blir mätproceduren ganska tidskrävande, vilket ger en ej försumbar CPU - belastning.
- Mätresultatet ger inga entydiga värden på belastningen utan endast antydningar om eventuell överbelastning.

Lösningsförslag enligt denna definition återfinnes i avsnitt 4.4.

3.4.2 Längden på kön i väntan på I/O - enhet.

Definition: Med längden på kön i väntan på I/O - enhet menas det antal program som väntar på tillgång till en viss I/O - enhet för in eller utläsning av information.

Motiv för användning: En upplysning om längden på kön till en viss I/O - enhet är värdefull då man önskar veta dess utnyttjandegrad. Speciellt gäller det enheter av typ skivminne, trumminne o.s.v. där det inte direkt syns när överföring av information sker. Därför blir mätning på kön till exempelvis skrivenheter av mindre intresse då det i sådana fall tydligt syns när maskinen arbetar.

Utdata: Antal program som väntar på viss I/O - enhet.

Fördelar: - Enkla och entydiga mätvärden ger enkel efterbehandling av resultatet.
- De enskilda programmens identitet behöver inte vara kända.
- Inskränkes mätningen till en kontroll om något program finns i kön eller om den är tom blir mätproceduren mycket enkel.

Nackdelar:- De erfordeliga uppgifterna finns i av operativsystemet kontrollerade områden, så vissa problem kan uppstå vid åtkomsten.
- Mätresultatet ger ingen entydig och klart definierad bild av utnyttjandegraden utan endast en indikation om eventuell överbelastning föreligger.

Lösningsförslag enligt denna defiition återfinnes i avsnitt 4.4.

3.4.3 Längden på kön i väntan på primärminne.

Definition: Med längden på kön i väntan på primärminnesutrymme menas det antal program som väntar på att få en plats tilldelad i primärminnet för att exekveras.

Motiv för användning: Upplysning om kön till primärminnet ger en uppfattning om dess storlek är tillräcklig i förhållande till de på systemet pålagda arbetsuppgifterna. Vidare kan icke-residenta programs medelresponstid i viss mån uppskattas.

Utdata: Antalet program som köar till primärminnet.

Fördelar: - Enkla och entydiga mätvärden ger enkel efterbehandling av resultatet.
- Enskilda programs identitet behöver ej vara kända.
- Begränsas mätningen till om något program köar blir blir mätproceduren mycket enkel.

Nackdelar:- Vissa manipulationer i operativsystemets arbetsareor är nödvändig för att få fram mätdata.
- Mätresultatet ger inga entydiga värden på belastningen utan endast antydningar om eventuell överbelastning.

Lösningsförslag enligt denna definition återfinnes i avsnitt 4.4.

3.5 Utnyttjandegraden av I/O - kanaler.

Definition: Med utnyttjandegraden av I/O - kanaler mesas den del av totalt förfluten tid som en viss I/O - kanal är upptagen.*)

*)Upptaget betyder i detta sammanhang att en mjuk eller hårdvarumässig flagga satts som anger att kanalen är reserverad för en specifik överföring.

Motiv för användning: Speciellt på den kanal som förbinder massminnet med centralenheten är en mätning av denna typ intressant. Detta därför att massminnets överföringskapacitet ofta utgör en begränsande faktor för ett datorsystems totala kapacitet.

Utdata: Beläggningsgraden på en viss I/O - kanal.

Fördelar: - Bara två tillstånd att hålla reda på, upptaget eller ledigt.
- Enkel form på mätvärdena ger okomplicerad efterbehandling av resultatet
- Ingen som helst kännedom om vad som överföres på kanalerna är nödvändig. Endast när n å g o n överföring sker.
- Sker mätningen hårdvarumässigt med inkoppling av yttre enhet belastas inte CPU överhuvudtaget.

Nackdelar:- Ger ingen information om CPU - aktivitet.
- Mätes tiden för upptaget respektive ledigtillstånd med den interna klockan kan problem med klockupplösningen uppstå.

- Delar flera I/O-enheter på samma kanal erhålles ingen information om den utnyttjandegrad som var och en enheterna ger upphov till.

Lösningsförslag enligt denna definition återfinnes i avsnitt 4.5.

K A P I T E L 4

LÖSNINGSFÖRSLAG

4. Lösningsförslag.

4.0 Inledning

I detta avsnitt kommer närmare att beskrivas ett antal lösningsförslag grupperade efter de i kapitel 3 redovisade definitionsgrunderna. Varje avsnitt inleds med en allmän beskrivning av den metodik som användes. Därefter följer ett eller flera lösningsförslag med angivande av positiva och negativa egenskaper.

En sammanställning över samtliga förslag följer nedan.

4.1 Exekveringstid

4.1.1 Mätning av total exekveringtid för operativsystem plus applikationsprogram.

4.1.1.1 Mätning av medelbelastningen under längre tid.

4.1.1.2 Mätning och notering av belastningen under kortare tidsintervall.

4.1.1.3 Mätning av belastningen styrd med hjälp av yttre enhet.

4.1.2 Mätning av exekveringstid för operativsystem respektive applikationsprogram.

4.1.2.1 Särskiljande av operativsystem och applikationsprogram genom tidmätning.

4.1.3 Mätning av exekveringstid för operativsystem respektive varje applikationsprogram för sig.

4.1.3.1 Tidmätning mellan uppstart och avslut av program.

4.1.3.2 Stickprovstagning på exekverande program vid klockbrytning.

4.1.3.3 Stickprovstagning på exekverande program initierad av yttre enhet.

4.1.4 Mätning av exekveringstiden för varje prioritetsnivå.

4.1.4.1 Tidmätning med notering av prioritetsnivå.

4.2 Exekveringsfrekvensen för varje prioritetsnivå.

4.2.1.1 Modifiering av STOP - rutinen.

4.2.1.2 Modifiering av varje program.

4.3 Primärminnesbeläggning

4.3.1.1 Direkt räkning av upptaget utrymme.

4.3.1.2 Kontroll av i primärminnet befintliga programs längd.

4.4 Kölängd

4.4.1.1 Kontroll om något köande program finns.

4.4.1.2 Mätning av antal köande program.

4.5 Utnyttjandegraden av I/O - kanaler.

4.5.1.1 Kontroll i operativsystemet av I/O - aktivitet.

4.5.1.2 Mätning av I/O -aktivitet med hjälp av yttre enhet.

4.1.1 Mätning av total exekveringstid för operativsystem plus applikationsprogram.

4.1.1.0 Allmänt

Principen är att använda ett enkelt program på lägsta prioritet, bestående enbart av en räknare (idle - loop). Detta program aktiveras uner den tid då inget annat med högre prioritet exekveras. På så sätt räknas inaktiv tid i CPU, d.v.s. den tid då varken operativsystem eller något applikationsprogram begär exekvering.

Total exekveringstid för operativsystem plus applikationsprogram enligt definitionen i 3.1.1 kan då erhållas såsom:

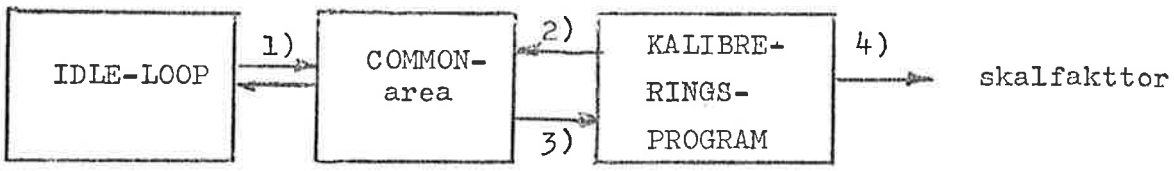
Total exekveringstid = total mättid - räknarens värde · ·progammets exekveringtid

Belastningens värde i procent blir då:

$$\text{Belastning} = \frac{\text{Total exekveringstid}}{\text{Total mättid}} \cdot 100 \quad (\%)$$

Räknarens värde, som mäste finnas tillgänglig i en area som kan nås av flera program, läses av med hjälp av ett speciellt avläsningsprogram. Detta program nollställer även räknaren. Två olika varianter av hur denna avläsning kan ske kommer att beskrivas, nämligen 4.1.1.1 och 4.1.1.2.

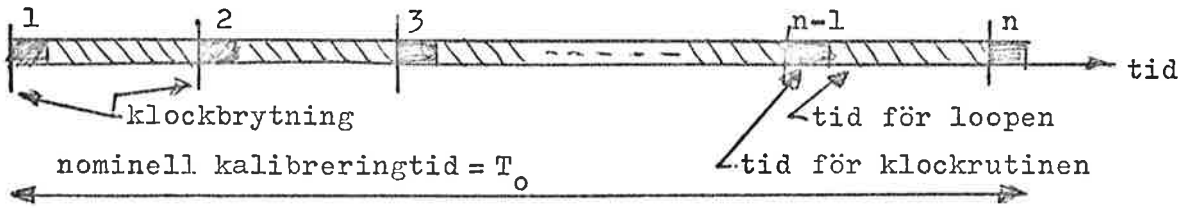
Eftersom idle - loopräknarens värde blir helt beroende av den aktuella datorn cykeltid och denna sällan är exakt angiven, måste loopen kalibreras. Denna kalibrering som antagligen endast behöver utföras en gång för varje datorsystem kan göras emot den interna klockan. Ett speciellt kalibreringsprogram aktiveras för att utföra kalibreringen. Programmet startar upp mätningen och nollställer loop-räknaren. Efter en viss fixerad tid läses loop-räknarens värde och en skalfaktor kan beräknas. Se figur 4.1.



Figur 4.1 Samspelet mellan idle-loopen, Common-arean och kalibreringsprogrammet. Siffrornas betydelse förklaras nedan.

1) Idle-loopen kommunicerar kontinuerligt med COMMON-arean och uppdaterar räknarens värde för varje varv som genomlöpes.
2) Vid första aktiveringen sker nollställning.
3) Vid andra aktiveringen sker avläsning.
4) Kalibreringsprogrammet producerar efter vissa enkla beräkningar som utresultat en skalfaktor.

När kalibrering sker får inga andra program vara aktiva eftersom detta skulle medföra att idle-loopen inte räknar under hela kalibreringstiden. I ett realtidssystem med regelbundna klockbrytningar blir så ändå inte fallet eftersom klockbrytningarna stjäl en del tid. Denna tid som tämligen exakt kan definieras ger, om den inte kan försummas, upphov till ett felaktigt värde på skalfaktorn. Se figur 4.2.



Figur 4.2 Aktiviteten i CPU under kalibreringen. Den verkliga kalibreringstiden fås såsom: $T = T_o - n \cdot$ (tid för klockrutin)

Eftersom det vidare tar en viss tid för uppstart och avslut av ett program måste, om en korrekt skalfaktor skall kunna beräknas, hänsyn tagas till detta. Även denna tid

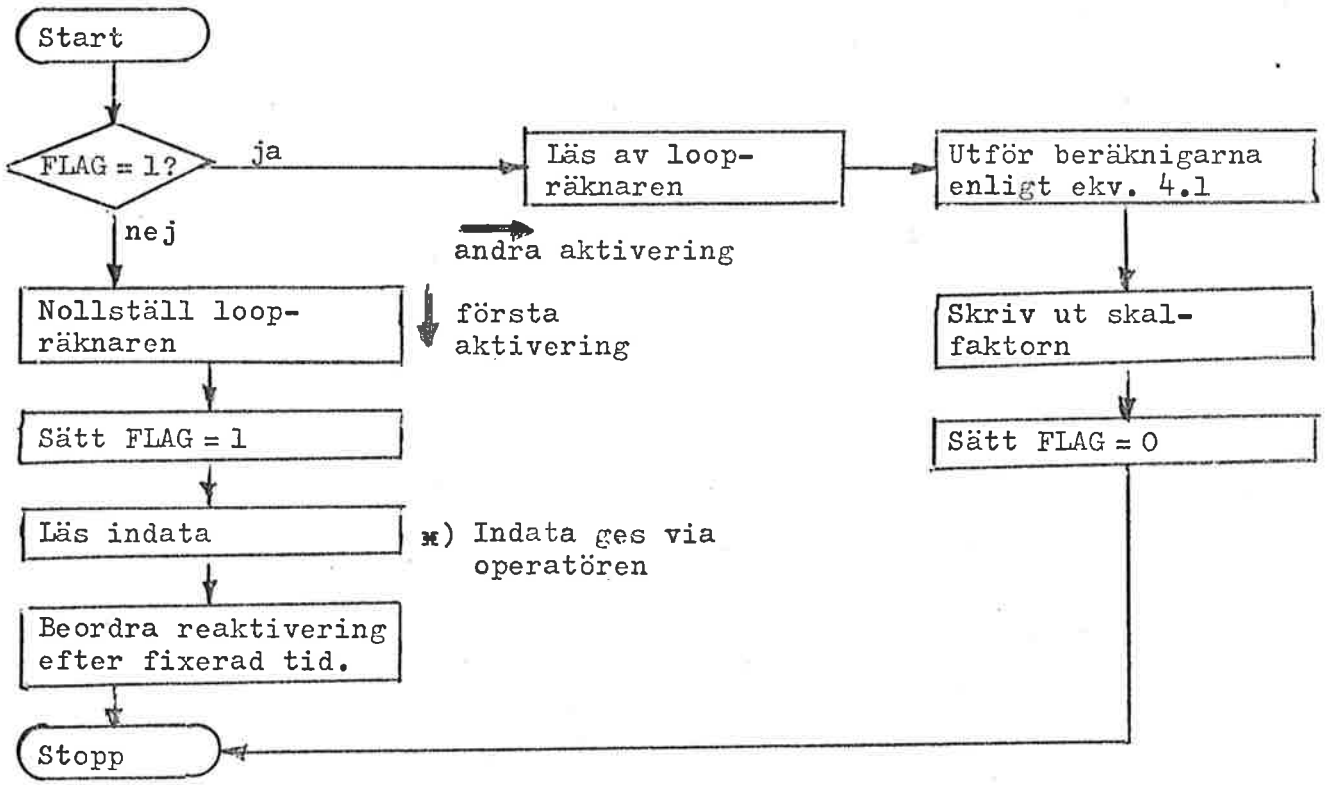
kan definieras relativt nogrannt och om den benämnes AD kan följande samband uppställas:

$$SK = \frac{T - n \cdot AD}{ILO} = \frac{T_o - n \cdot (AD + KR)}{ILO} \qquad \text{(tidsenheter/räknarsteg)} \qquad (4.1)$$

Där beteckningarna har följande betydelse:

ILO = Idle-loopräknarens värde
SK = Skalfaktor
T = Verklig kalibreringstid
n = Antal klockbrytningar under kalibreringstiden
T_o = Nominell kalibreringstid
KR = Tid för klockrutinen

Ett flödesschema över ett program som utför nödvändiga åtgärder finns i figur 4.3.



Figur 4.3 Kalibreringsprogram. Flag skall vara nollställd ifrån början.

När sedan mätningen startar blir det erhållna idle-loopvärdet multiplicerat med skalfaktorn inte ett helt korrekt värde för inaktiv tid (den tid då varken operativsystem eller något applikationsprogram önskar exekvering). Anledningen är att hänsyn även här måste tas till tiden för uppstart och avslut vid avbrott. Då antalet avbrott av loopen ej är känt kan endast en approximativ korrektion göras.

Om det antages att loopen k a n startas endast en gång per klockbrytningsintervall kan uppställningen enligt figur 4.4 göras.

Uträknad belastning utan korrektion	0 %	25 %	50 %	75 %	100 %
Antal gånger loopen kan inkallas i förhållande till totala antalet klockbrytningar under mätintervallet	100% ↕ 100%	100% ↕ 75%	100% ↕ 50%	100% ↕ 25%	100% ↕ 0%
Medelvärde av max och min	100%	87.5%	75.0%	62.5%	0%

Figur 4.4 Exempel vid olika belastningar visande hur stor del av totala antalet klockbrytningar som idle-loopprogrammet kan inkallas, vid antagandet enligt texten.

Om medelvärdena enligt figur 4.4 plottas som funktion av uträknad belastning utan korrektion erhålles en kurva M som i figur 4.5.

Medelvärdet M i figur 4.5 fungerar emellertid dåligt som grund för ett lämpligt korrektionsvärde på grund av dess beteende vid höga belastningar. Notera speciellt diskontinuiteten vid 100 %. En lämpligare kurva bör intuitivt vara N som, om belastningen utan korrektion benämnes B^*, kan uttryckas matematiskt som :

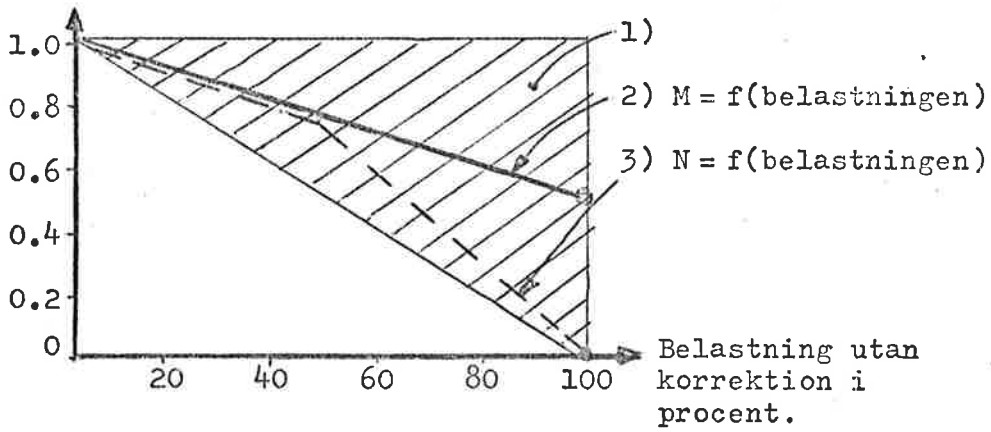
$$N = \begin{cases} 1 - \frac{B^*}{200} & 0 \leq B^* \leq 50 \quad \text{(procent)} \\ 1.5 - \frac{B^*}{100} \cdot 1.5 & 50 \leq B^* \leq 100 \quad \text{(procent)} \end{cases} \qquad (4.2)$$

Användes sedan ekvation 4.2 som korrektionsfaktor, för att approximativt korrigera för idle-loopens tid för uppstart och avslut, kan ett bättre värde på belastningen erhållas som ekvation 4.3. (KORR = N enligt ovan)

$$B = 100 - \frac{ILO \cdot SK + n \cdot KORR \cdot AD}{\text{Mättidens längd}} \cdot 100 \quad \text{(procent)} \qquad (4.3)$$

Det är mycket möjligt att de gjorda korrektionerna för tidsförlust vid avbrott i praktiken kan uteslutas. Anledningen är att storheterna KR och AD, eventuellt är försumbart små i förhållande till andra tider. Om så är fallet kan uttrycket enligt ekvation 4.4 användas istället.

Förhållande mellan antalet idle-loop-inkallningar och totala antalet klockbrytningar.

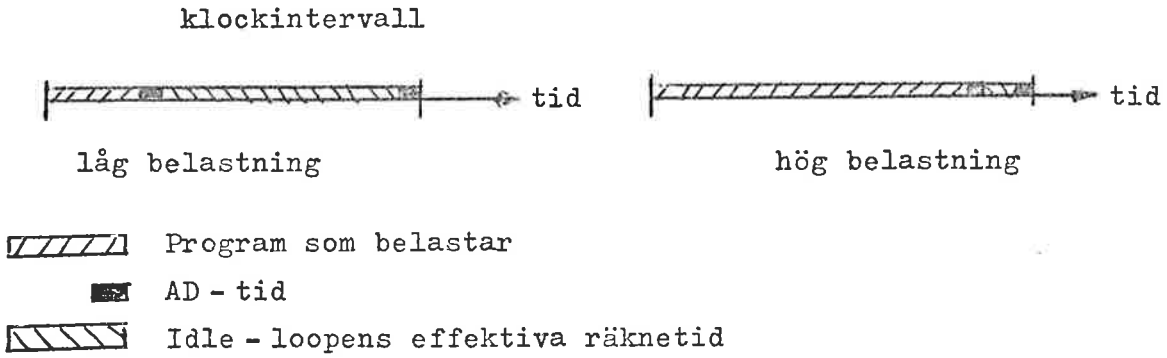


Figur 4.5 1) det streckade området anger möjlig variation enligt figur 4.4.
2) Medelvärde enligt figur 4.4.
3) Den använda korrektionskurvan.

$$B = 100 - \frac{ILO \cdot SK}{\text{mättidens längd}} \quad \text{(procent)} \qquad (4.4)$$

(skalfaktorn SK okorrigerad)

Kriteriet för vid vilka värden på AD respektive KR dessa kan försummas beror givetvis på uppställda noggrannhetskrav. Generellt bör en gräns på storleksordningen några procent av klockintervallet vara lämpligt. Försiktighet måste emellertid iaktagas vad det gäller AD. Vid belastningar nära 100 procent blir dess storlek nämligen stor i förhållande till idle-loopens effektiva räknetid, även om dess absoluta tid är liten. Se figur 4.6.



Figur 4.6 Illustation av tidsfördelningen i ett klockpulsintervall vid olika belastning.

Normalt ger mätningar av total exekveringstid på detta sätt med idle-loop ingen information om varje enskilt applikationsprograms CPU - belastning. Denna kan emellertid erhållas med programvis inkoppling, kombinerat med en mätning då samtliga program är aktiva. Sedan kan respektive programs totala exekveringstid, under mättiden, beräknas med ekvation 4.5.

$$P(n) = \text{deltid}(n) - \frac{1}{N-1} \cdot \left(\sum_{k=1}^{N} \text{deltid}(k) - \text{totaltid} \right) \qquad (4.5)$$

P(n) = n:te programmets tidsbelastning.
deltid(n) = Erhållen tidsbelastning med enbart program n aktivt.
N = Totala antalet program som mätningen utföres på.
Totaltid = Erhållen tidsbelastning med alla program aktiva.

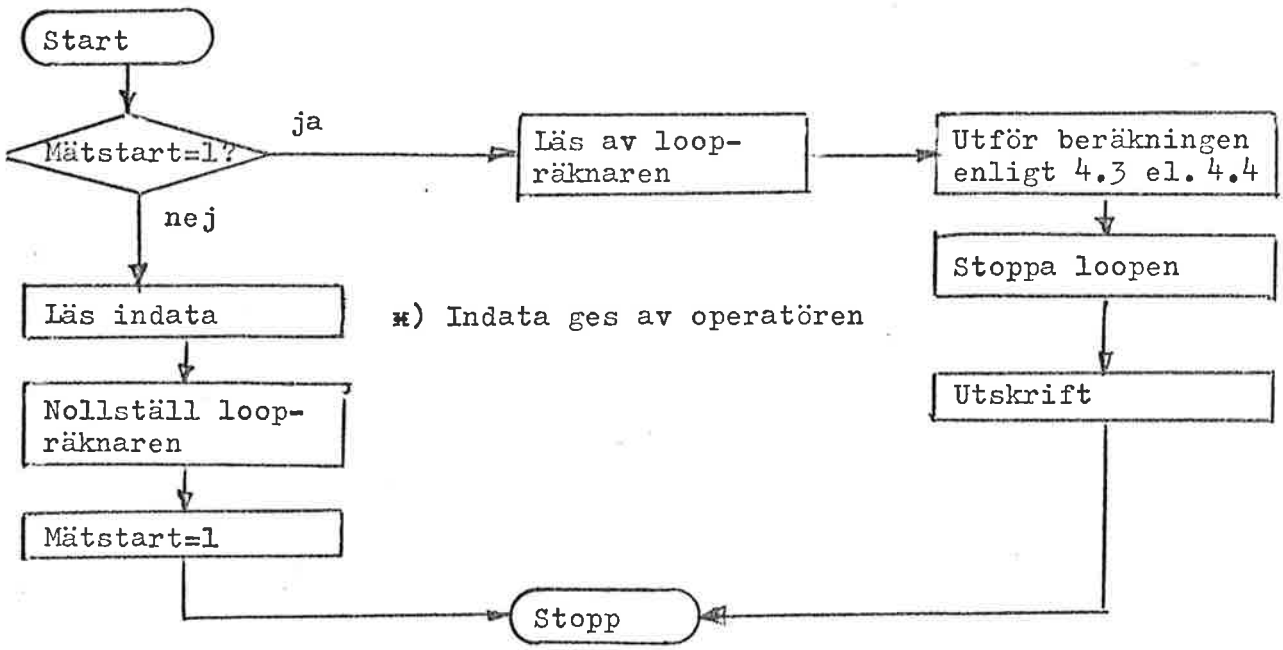
En varning är befogad vid användning av denna mätmetodik. Dels kommer inte alla de av applikationsprogammet utnyttjade rutinerna i operativsystemet med i resultatet, sefigur 3.13 på sidan 3.20. Dessutom kan, när ett program ensamt har tillträde till centralenheten, lätt en skev bild av den verkliga belastningssituationen uppstå.

4.1.1.1 Mätning av medelbelastningen under längre tid.

Idle-loopen nollställes och tillåts sedan under en längre tid, storleksordningen minuter , räkna innan avläsning sker.

När kalibrering genomförts och mätning skall startas aktiveras det program som styr mätningen. Vid aktiveringen anges också när reaktivering skall ske för avläsning av loopen och beräkning av belastningen. Tiden mellan aktiveringarna blir då lika med nominella mättiden .

Ett progam som styr idle-loopen och utför beräkningarna enligt ekvation 4.3 eller 4.4 finns beskrivet i figur 4.8.



Figur 4.8 Mätprogram. Mätstart skall ha värdet noll ifrån början.

De utdata som kan erhållas blir enbart medelvärdet av belastningen under längre tid enligt 3.0.1 A sidan 3.4.

—— Positiva egenskaper hos metoden:

- Enkla program att skriva. Kan utan problem utföras i högnivåspråk, exempelvis FORTRAN.
- Lätt att anpassa metoden till olika datorer, endast vissa parametrar behöver ändras plus vissa maskinberoende kommandon.
- Utdata, medelbelastningen är enkel att tolka.

—— Negativa egenskaper hos metoden:

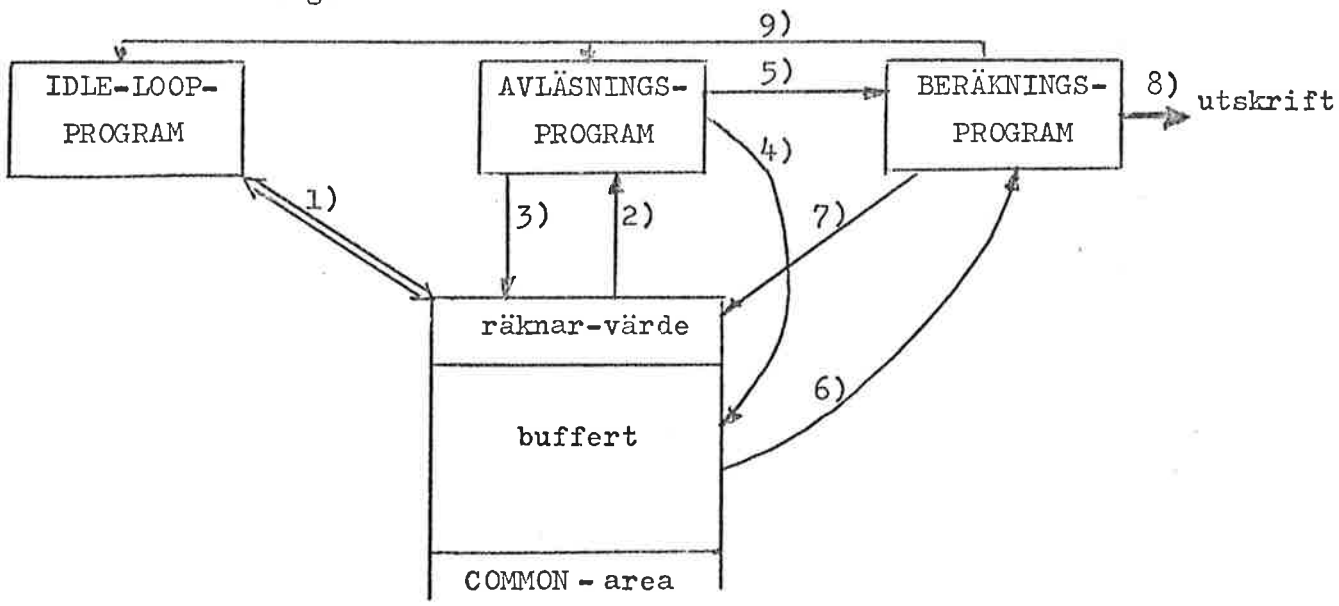
- Beroendet av datorns cykeltid och därför nödvändig kalibrering ger extraarbete.
- Enbart medelbelastningen under längre tid som utdata. Därför små möjligheter att följa belastningens variationer.

4.1.1.2 Mätning och notering av belastningen under kortare tid.

Denna metod är mycket lik den föregående, en idle-loop av samma typ användes. Skillnaden består i att avläsning och nollställning av loop-räknaren sker betydligt oftare med lagring av varje värde, för vidare bearbetning senare.

Avläsningsintensiteten kan enkelt varieras med operatörskommando ifrån maximalt en avläsning vid varje klockbrytning till i princip hur sällan som helst. Antalet mätvärden beror naturligtvis på avläsningsintensiteten, men normalt bör dess antal bli så stort att någon form av undanlagring måste ske före bearbetning. Lämpligast löses det med en buffert som tar hand om mätvärdena allt eftersom dessa noteras. När bufferten är fylld töms den av ett speciellt beräkningsprogram som sorterar och utför nödvändiga beräkningar. Vid denna tömning bör mätningen tillfälligt avbrytas för att återupptagas när bufferten är tömd.

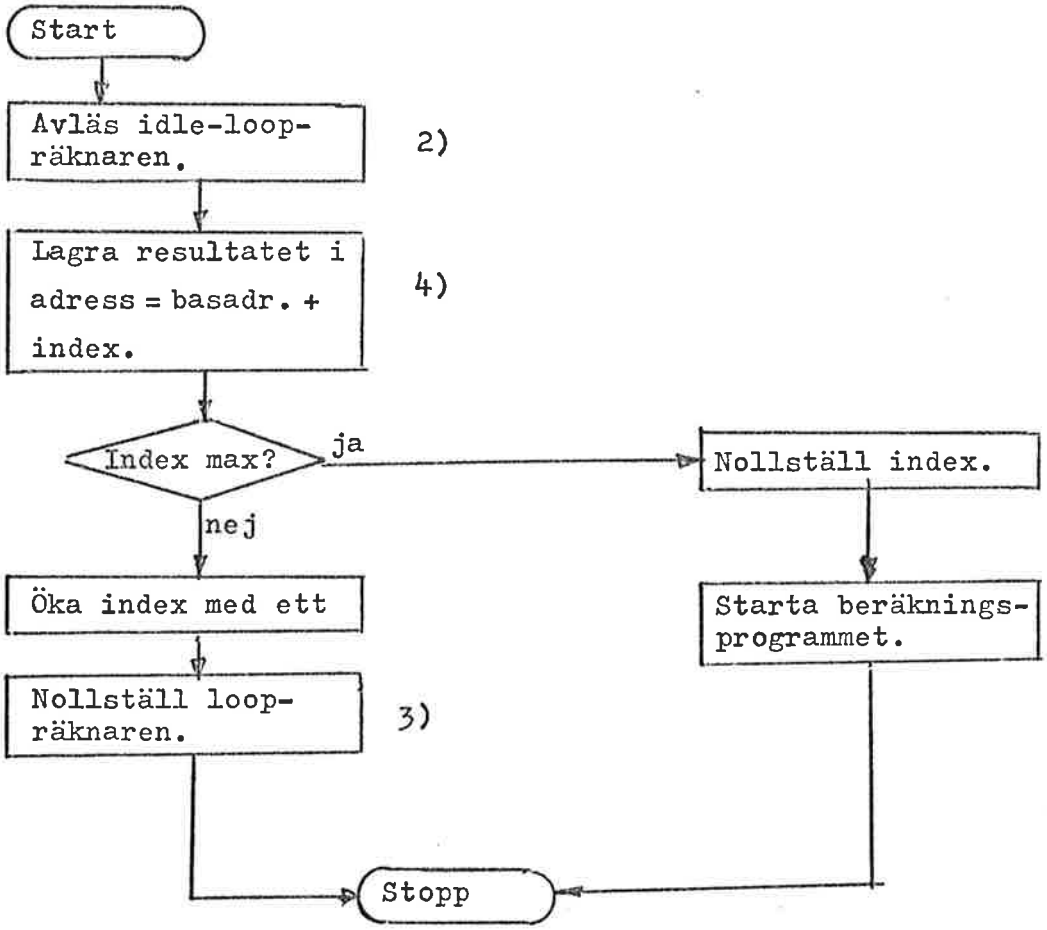
För att utföra det nödvändiga arbetet behövs flera olika program och en uppdelning enligt figur 4.9 kan exempelvis användas. Där användes tre program, idle-loop, avläsningsprogram med buffert och beräkningsprogram. Dessa kommunicerar med varandra via en global COMMON-area för vidarebefordran av mätvärden och nollställning.



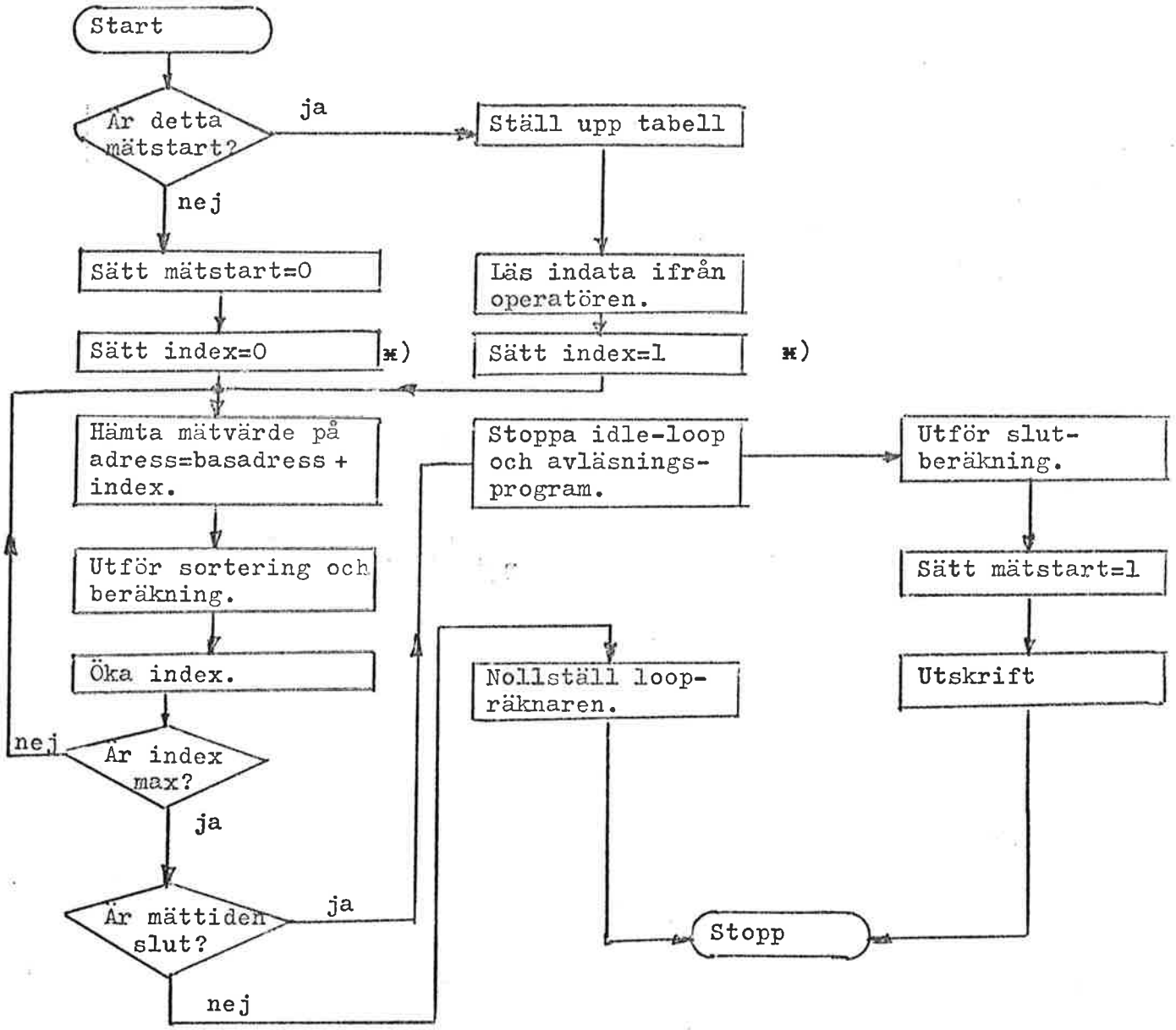
Figur 4.9 Figur som visar kommunikationen mellan de tre programmen. Siffrornas betydelse förklaras nedan:

1) Idle-loopen uppdaterar kontinuerligt räknar-värdet i COMMON-arean.
2) När aktivering sker avläses räknarens värde.
3) Nollställning av räknaren.
4) Räknarens värde lagras i bufferten.
5) När bufferten är fylld startar beräkningsprogrammet.
6) Beräkningsprogrammet tömmer bufferten.
7) Nollställning av räknaren som talar om att den avbrutna mätningen åter kan starta.
8) Utskrift sker efter nödvändiga beräkningar.
9) Beräkningsprogrammet stoppar idle-loopprogrammet och avläsningsprogrammet vid mättidens slut.

Avläsningsprogrammets principiella utseende framgår av figur 4.10 och beräkningsprogrammet av figur 4.11.



Figur 4.10 Avläsningsprogrammets utformning. Siffrorna refererar till motsvarande angivna i figur 4.9.



Figur 4.11 Beräkningsprogrammets uppbyggnad.

*) Det allra första mätvärdet ignoreras eftersom loopens räknetid ej är känd första gången.

Utdatas form kan med denna metod väljas inom vida gränser. Samtliga de i avsnitt 3.0.1 redovisade presentationssätten är möjliga att använda. Skillnaden i mjukvaruutformningen för de olika sätten inskränker sig till vissa beräknings och utskriftsrutiner.

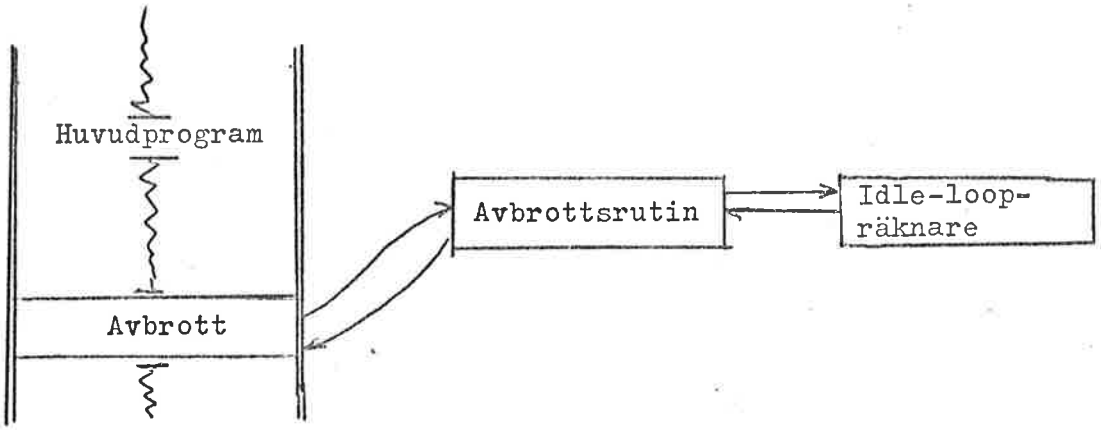
—— Positiva egenskaper hos metoden:
- Programmen som behövs kan skrivas i högnivåspråk, vilket underlättar arbetet.
- Lätt att anpassa till olika datorer. Endast vissa parametrar och maskinberoende kommandon behöver ändras.
- Stor flexibilitet vid val av utdatas form.
- Enkelt att ändra mätregistreringarnas frekvens-

—— Negativa egenskaper hos metoden:
- Beroendet av datorns cykeltid med därför nödvändig kalibrering ger extraarbete.

4.1.1.3 Mätning av belastningen styrd med hjälp av yttre enhet.

Det som skiljer denna metod ifrån den föregående är att i stället för att använda ett avläsningsprogram som styrs av operativsystemet, ansluts en yttre enhet för denna styrning. Den yttre enheten består av en pulsgenerator som genererar pulser av lämplig amplitud och längd. Pulserna alstrar avbrottssignaler som ger uthopp till en avbrottsrutin med avläsningsprogram.

Samma typ av idle-loop som tidigare kan användas. Denna loop kan eventuellt vara aktiv för jämnan, eller åtminstone under den tid då mätning kan bedömas bli befogad. Fördelen med detta är att mätproceduren i så fall snabbt kan startas upp genom att starta pulsgeneratorn, utan några operatörskommandon via operativsystemet. Avbrottsrutinen blir nämligen direkt aktiverad när avbrottspuls kommer. Se figur 4.12. Mätningen kan sedan avslutas med ny avbrottspuls



Figur 4.12 När avbrottssignalen kommer sker uthopp till avbrottsrutinen som innehåller avläsningsprogrammet. Detta avläser idle-loopräknarens värde.

Avbrottspulsernas frekvens kan enkelt justeras på pulsgeneratorn och blir helt oberoende av datorns interna klocka. Detta ger stora möjligheter att välja önskat mätinervall, som även kan vara kortare än ett klockpulsintervall.

Avläsningsprogrammet kan ha exakt samma uppbyggnad som det i figur 4.10 tidigare visade. Över beräkningsprogrammet som tar hand om de insamlade mätvärdena och bearbetar dessa kan i princip utformas som i föregående avsnitt. Dock bör utskriftsdelen utformas som ett separat program med manuell aktivering. Orsaken är att utskrift ej sker automatiskt vid mättidens slut eftersom denna styrs av operatören själv utan kommunikation med operativsystemet.

Eftersom avbrottsrutinen alltid har högre prioritet än det program som utför beräkningarna kan bufferten för mätvärdena eventuellt börja fyllas på nytt innan den är tömd. En viss försiktighet måste emellertid iaktagas vid hög mätfrekvens så att bufferten inte fylls i snabbare takt ån den töms. Vid rimliga värden på mätfrekvensen, alltså pulsgeneratorns frekvens, bör detta dock inte ställa till några problem.

Även utdatas form kan i lika hög grad som i tidigare avsnitt varieras på ett enkelt sätt.

—— Positiva egenskaper hos metoden:

- Möjlighet att snabbt och enkelt starta mätförloppet.
- Enkelt att ändra mätregistreringarnas frekvens.
- Stor flexibilitet vid val av utdatas form.

—— Negativa egenskaper hos metoden:

- Kräver anslutning av yttre enhet.
- Kravet på kalibrering av datorns cykeltid ger extra-arbete.

4.1.2 Mätning av exekveringstid för operativsystem respektive applikationsprogam.

4.1.2.0 Allmänt

Denna mätform är en utveckling av 4.1.1 , som ger mer differentierad information genom särskillnad av operativsystem och applikationsprogram och deras tidsbelastning på CPU. En idle-loop enligt tidigare användes för att mäta outnyttjad tid i CPU. Därav följer krav på kalibrering av cykeltiden för loopen. Hur detta sker utredes på sidorna 4.3 - 4.5.

Det som tillkommer jämfört med den i föregående avsnitt beskrivna mätmetoden är en sekvens som kontrollerar vad för slags arbete som pågår i centralenheten. Administrerande funktioner av operativsystemet eller användarinriktat av applikationsprogrammen. Här måste, såsom nämts i samband med definitionen, alla rutiner såsom I/O-rutiner, biblioteksrutiner o.s.v. vars uppgift är att komplettera användarprogrammen, även räknas till dessa.

Svårigheter kan eventuellt uppstå när det gäller att särskilja dessa speciella rutiners exekvering ifrån övriga i operativsystemet. En kontroll som kan användas är att se efter vilket program som anges som aktivt. Även när I/O-utmatning sker med därför avsedda drivprogram står programmet

som utnyttjar dessa kvar som aktivt.

Problem uppstår vid avbrott som hör samman med andra aktiviteter än de pågående. Därför måste vid avbrott undersökas dess typ och om det kan anses höra till det för tillfället aktiva programmet.

Alla dessa åtgärder blir relativt komplicerade och kräver säkerligen en hel del tid. Tid som ger en ökad belastning på centralenheten. Beroende på operativsystemets uppbyggnad blir lösningen helt olika till olika system. Generellt bör en tidsåtgång på storleksordningen 100 microsekunder kunna tillåtas, vilket säkert kan vara svårt att uppfylla i många fall.

Eftersom mätningen är uppdelad på två fristående delar, idle-loop och rutinen ovan, krävs en sammanställning av de bägge. Ligger idle-loopen som ett separat program måste dess belastning subtraheras ifrån totalt uppmätt. Ekvation 4.6 och figur 4.13 illustrerar detta.

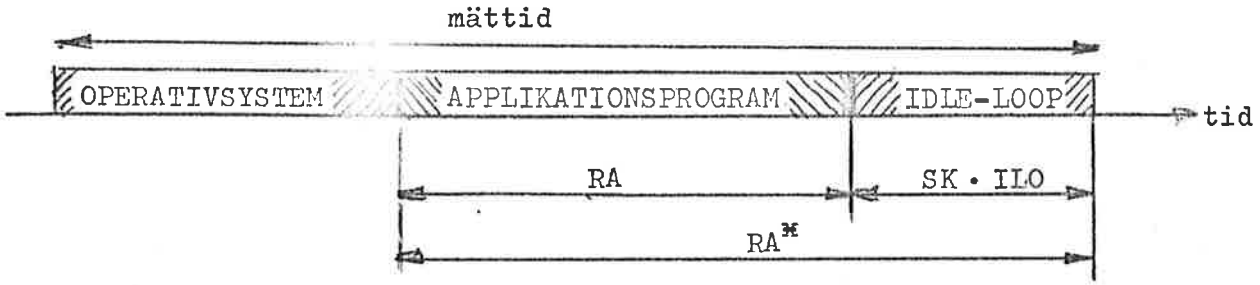
$$RA = RA^{*} - SK \cdot ILO \qquad (4.6)$$

RA = Den sanna utnyttjandetiden.

RA^{*} = Den av mätrutinen uppmätta utnyttjandetiden av CPU för applikationsprogrammen.

SK = Skalfaktor (se sidan 4.6)

ILO = Idle-loopräknarens värde.



Figur 4.13 Illustration till ekvation 4.6.

Åtminstone två sätt att mäta den tid som operativsystemet respektive applikationsprogram använder centralenheten är möjliga. Det ena är att direkt mäta de deltider som respektive aktivitet kräver. Tidmätare kan endera vara den interna , eller också en yttre ansluten klocka. Fördelen med det senare är att klockans upplösning i princip kan göras hur fin som helst. Interna klockans minsta tidsenhet är kanske inte tillräckligt liten för tillfredsställande noggrannhet.

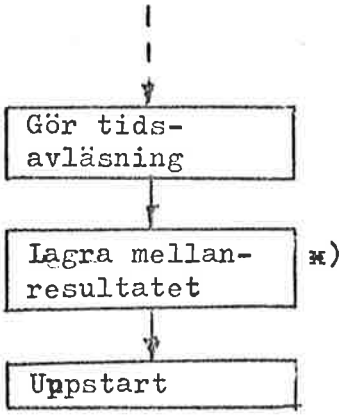
Det andra mätsättet är att med vissa mellanrum, av konstant eller slumpmässig längd, ta stickprov på vilken typ av aktivitet som försiggår i CPU. Resultatet behandlas sedan statistiskt enligt de i kapitel 3 beskrivna metoderna.

Här kommer endast en metod med direkt tidmätning att beskrivas.

4.1.2.1 Särskiljande av operativsystem och applikationsprogram genom tidmätning.

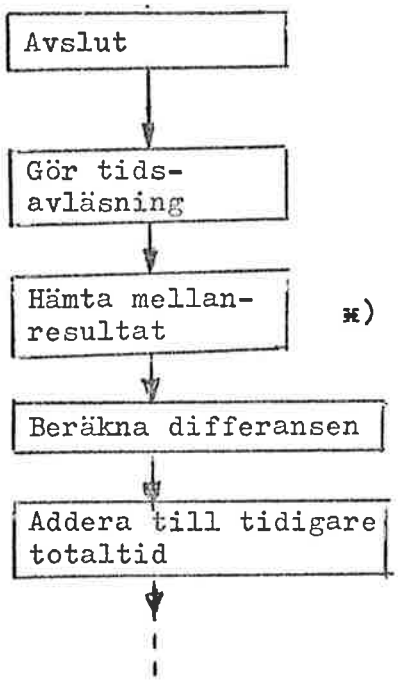
Vid uppstart av ett applikationsprogram läses en klocka av och tiden lagras. När sedan progammet exekverats färdigt sker en ny klockavläsning och utnyttjad tid beräknas. Kommer ett avbrott måste dess typ kontrolleras. Huruvida det är ett avbrott orsakat, direkt eller indirekt, av programmet själv eller av annan händelse. Om orsaken orsaken visar sig oberoende av den pågående aktiviteten i centralenheten skall tidmätningen tillfälligt avbrytas, annars inte.

Eftersom när program av en eller annan anledning lämnas, uthopp kan ske till olika ställen i operativsystemet måste rutiner infogas på flera ställen . Dessa rutiner får olika utseende beroende på anledningen till uthoppet. Sålunda kan utformningen vid uppstart se ut som i figur 4.14 och vid avslut enligt figur 4.15. Utseendet vid avbrott blir något mer komplicerat och framgår av figur 4.16.

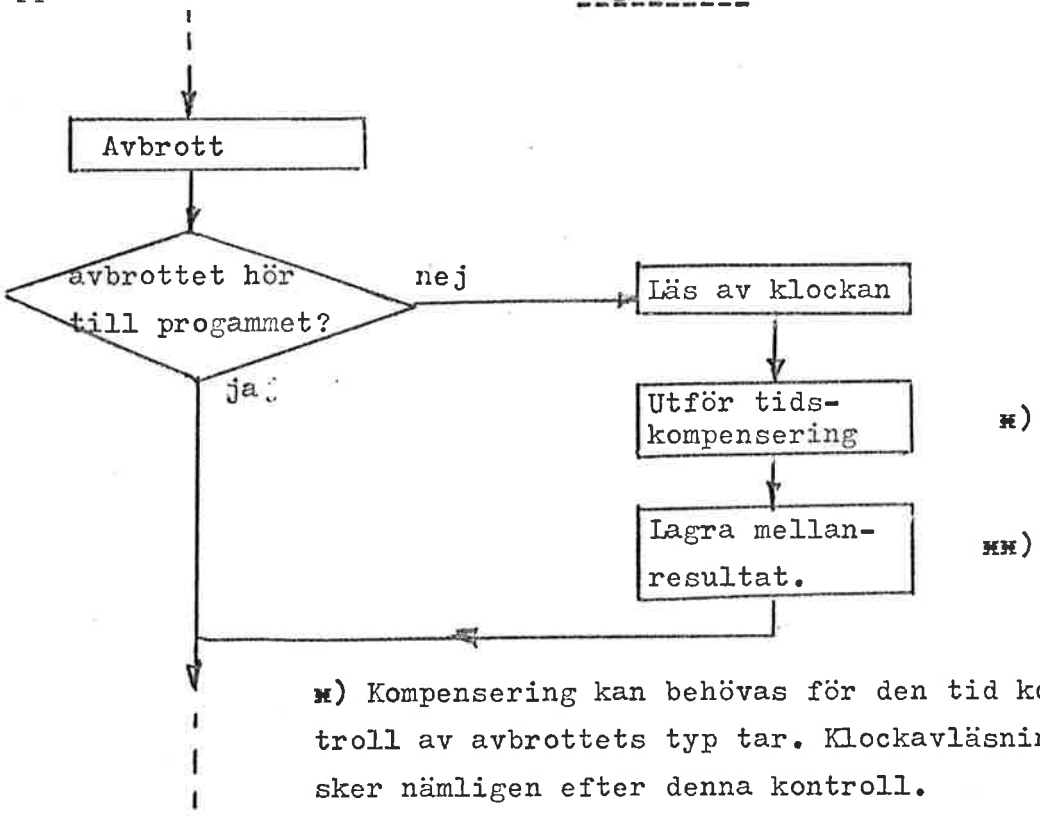


*) Samma värde i både uppstart och avslutsrutinen. Detta lagringsutrymme användes även av rutinen i figur 4.16.

Figur 4.14 Uppstartsrutin



Figur 4.15 Avslutsrutin



*) Kompensering kan behövas för den tid kontroll av avbrottets typ tar. Klockavläsningen sker nämligen efter denna kontroll.

**) Se kommentaren i figur 4.14.

Figur 4.16 Komplettering till avbrottsrutinensbörjan. Rutinen i slutet kan ha samma utseende som den i figur 4.15.

De gjorda avläsningarna i de olika rutinerna måste lagras så att gemensam åtkomst är möjlig. Detta eftersom det är samma tid som mätes i samtliga fall.

Ett speciellt avläsningsprogram inkallas med jämna mellanrum för att läsa av mätresultatet ifrån både tidmätningen och idle-loopräknaren. Resultatet lagras i en buffert som töms och bearbetning sker med hjälp av ytterligare ett program, kallat beräkningsprogram. Vid varje avläsning skall de båda lagringsutrymmena nollställas. Utseendet på de båda programmen överenstämmer i stort med de på sidorna 4.12 och 4.13 beskrivna. Skillnaden inskränker sig till att de båda mätvärdena hela tiden bearbetas parallellt för att vid slutberäkningen jämkas samman.

Idle-loopmätningen ger efter vissa enkla beräkningar ett mått på den tid då varken operativsystemet eller applikationsprogrammen använder CPU. Tidmätningen anger, efter korrektion enligt ekvation 4.6 sidan 4.17, applikationsprogrammens totala tidsbelastning av centralenheten. Med kännedom om totala mätiden kan sedan respektive belastningar beräknas enligt ekvationerna 4.7 och 4.8.

$$RA = \frac{RA^{*} - SK \cdot ILO}{\text{total mätid}} \cdot 100 \qquad \text{(procent)} \qquad (4.7)$$

(beteckningar enligt tidigare)

$$RO = 100 - \frac{RA + SK \cdot ILO}{\text{total mätid}} \cdot 100 \qquad \text{(procent)} \qquad (4.8)$$

RO = Operativsystemets procentuella belastning.

Stora variationsmöjligheter finns vid presentationen av resultatet. De i avsnitt 3.0.1 beskrivna metoderna är samtliga direkt applicerbara.

— Positiva egenskaper hos metoden:
- Stora möjligheter till variation av utdatas form.
- Enkelt att ändra tidsavståndet mellan mätregistreringarna.

— Negativa egenskaper hos metoden:
- Ändringar och tillägg till operativsystemet ofrånkomliga.
- De tillägg som behöver göras ökar belastningen och gör systemet långsammare.
- Idle-loopens beroende av cykeltiden medför nödvändig kalibrering.

4.1.3 Mätning ev exekveringstid för operativsystem respektive varje applikationsprogram var för sig.

4.1.3.0 Allmänt

Då man önskar erhålla varje enskilt programs belastning måste möjlighet finnas att identifiera alla i systemet befintliga program. En tabell över dessa behöver byggas innan mätningen startar, om inte utrymme redan finns i befintlig sådan. Den tabell där utrymme eventuellt kan finnas är den sammanställning där för samtliga program specificeras uppgifter som namn, prioritet, minneslokalisering o.s.v. Det troliga är emellertid att en ny tabell behöver sammanställas. I så fall hämtas uppgifter till denna i den ovan nämda sammanställningen över programmen.

Metodiken i de två fallen, med och utan befintligt tabellutrymme beskrives närmare i de följande avsnitten A och B.

När det gäller att läsa och skriva i av operativsystemet kontrollerade areor kan eventuellt problem med minnesskydd uppstå. Problemets svårighetsgrad beror emellertid helt på det aktuella systemets uppbyggnad och allmängiltiga synpunkter är svårt att ge. I de flesta fall bör dock svårigheten kunna övervinnas ganska enkelt.

Mätmetodens utformning liknar till stor del den i avsnitt 4.1.2 beskrivna. Det som tillkommer är endast byggandet av tabellen ovan och att vid varje mättillfälle kontrollera exekverande programs identitet. Samma svårigheter som tidigare uppstår vid avgörandet av vilka av operativsystemets rutiner som skall räknas till applikationsprogrammen.

Mängden mätdata blir stor och tiden för sortering och beräkning kan lätt ge upphov till en icke oväsentig belastning av centralenheten. Därför måste denna del mätoperationen hållas under uppsikt för att undvika systematiska fel.

A. Lagring av mätvärdena i befintlig tabell.

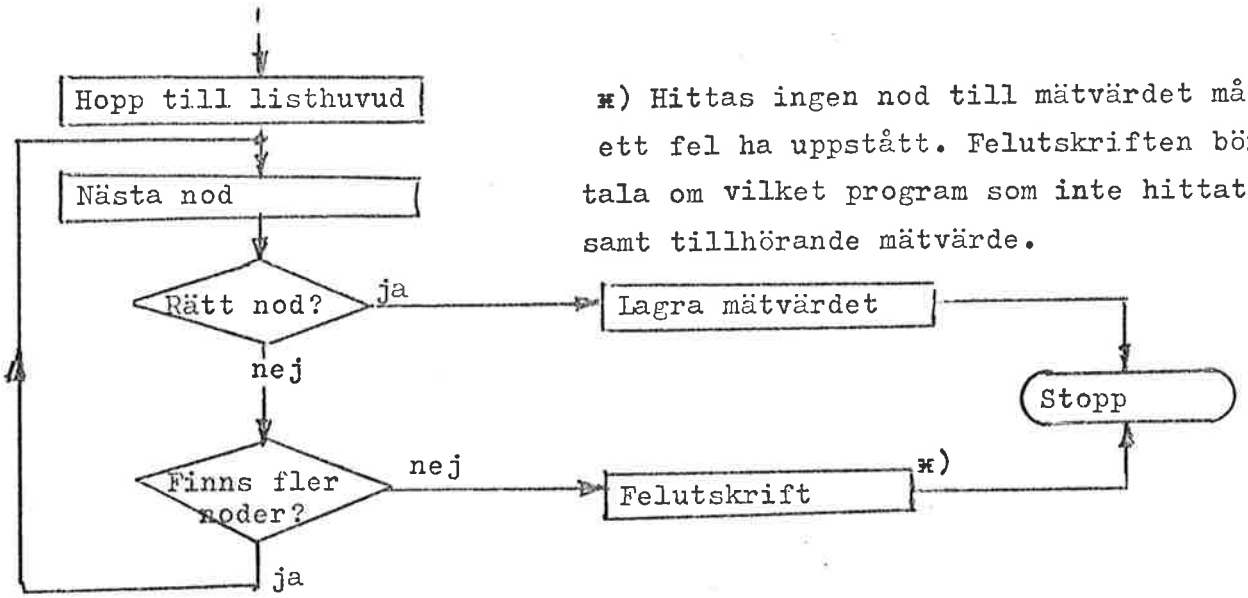
När ledigt tabellutrymme finns i befintlig tabell behöver mätningen inte inledas med uppställande av ny tabell. Däremot måste alla aktuella värden nollställas före mätstart.

När ett mätvärde registrerats skall detta sorteras in i tabellen på rätt plats. Detta kan, om många program finns i systemet, ta ganska lång tid i anspråk. Om tiden bedöms vara för lång kan en buffert användas där alla värden dumpas för vidare bearbetning senare. Programmens identitet måste dock fastställas och lagras tillsammans med respektive mätvärde.

Även när buffert användes måste antagligen sortering med jämna mellanrum tillgripas eftersom buffertens fysiska dimensioner måste begränsas. Vid denna sortering, som utföres av ett separat sorteringsprogram , bör mätningen tillfälligt avbrytas. Det nödvändiga arbetet tar nämligen ganska lång tid och belastningen på centralenheten ökar. Felaktiga resultat skulle alltså bli följden om mätningen pågick kontinuerligt. Visserligen missas ett eller flera värden, men

detta inverkar mindre på slutresultatet, om någon form av medelvärdesbildning användes, än ett eller flera för höga belastningsvärden.

Beroende på typen av tabell blir sorteringsmetodiken något olika. Ett exempel vid användning av länkad lista visas i figur 4.17.



*) Hittas ingen nod till mätvärdet måste ett fel ha uppstått. Felutskriften bör tala om vilket program som inte hittats samt tillhörande mätvärde.

Figur 4.17 Programavsnitt som söker efter rätt tabellplats.

B. Lagring av mätvärdena i nybyggd tabell.

När en ny tabell behöversammanställas undersökes vilka program som finns i systemet. Dessa uppgifter kan som redan nämnts erhållas ifrån den lista i operativsystemet som specificerar varje program. Utseendet på lämplig tabell framgår av figur 4.18

Programnamn
Mätdata
Programnamn
Mätdata
Programnamn

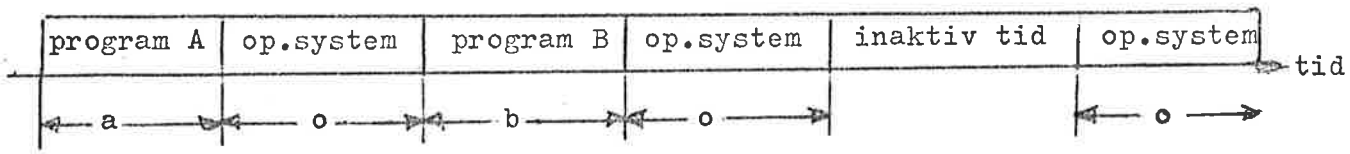
Figur 4.18 Lämpligt utseende på den tabell som skall uppställas innan mätningen startar.

Samma typ av buffert som i avsnitt A bör införas för lagring av mätvärdena innan sortering. Sorteringsförfarandet blir inte riktigt lika det i figur 4.17 visade eftersom den förutsätter tabell av typen länkad lista. Skillnaden blir, att med tabell enligt 4.18 ges nästa lagringsställe automatiskt eftersom lagringen sker konsekutivt. Vid länkad lista ges nästa adress i nodhuvudet.

De storheter som skall mätas är, som framgår av figur 4.19, operativsystemets respektive varje applikationsprograms exekveringstid. Denna mätning kan uppdelas i två delar. Dels en mätning av inaktiv CPU - tid, enligt tidigare beskrivna metoder, med en idle-loop. Dessutom en rutin som mäter varje applikationsprograms exekveringstid. Två principer kan användas för detta:

- A) Direkt tidtagning under exekveringen.
- B) Stickprov med kontroll av vilket program som exekveras.

Princip A användes i lösningsförslag 4.1.3.1 och B i 4.1.3.2 samt 4.1.3.3.



Figur 4.19 De storheter som skall mätas är a, b och o.

Direkt tidtagning har fördelen att ge sanna värden på exekveringstiden, men kräver omfattande arbete under hela mättiden med avläsning av tider. Stickprov ger betydligt enklare mätprocedur, men ett omfattande arbete i form av statistikberäkningar i efterhand krävs. Dessutom blir

stickprov alltid behäftade med en viss osäkerhet. Signifikansen måste beräknas.

Genom mätning av de två storheterna, inaktiv tid med idle-loopen och varje programs exekveringstid kan belastningen beräknas enligt ekvationerna 4.9 och 4.10.

$$RO = \left(1 - \frac{SK \cdot ILO + \sum RA^{*}}{\text{total mättid}} \right) \cdot 100 \qquad \text{(procent)} \qquad (4.9)$$

RO = Operativsystemets procentuella belastning.
SK = Kalibreringsfaktor för idle-loopen.
ILO = Idle-loopräknarens värde.
$\sum RA^{*}$ = Summan av alla applikationsprograms exekveringstid.

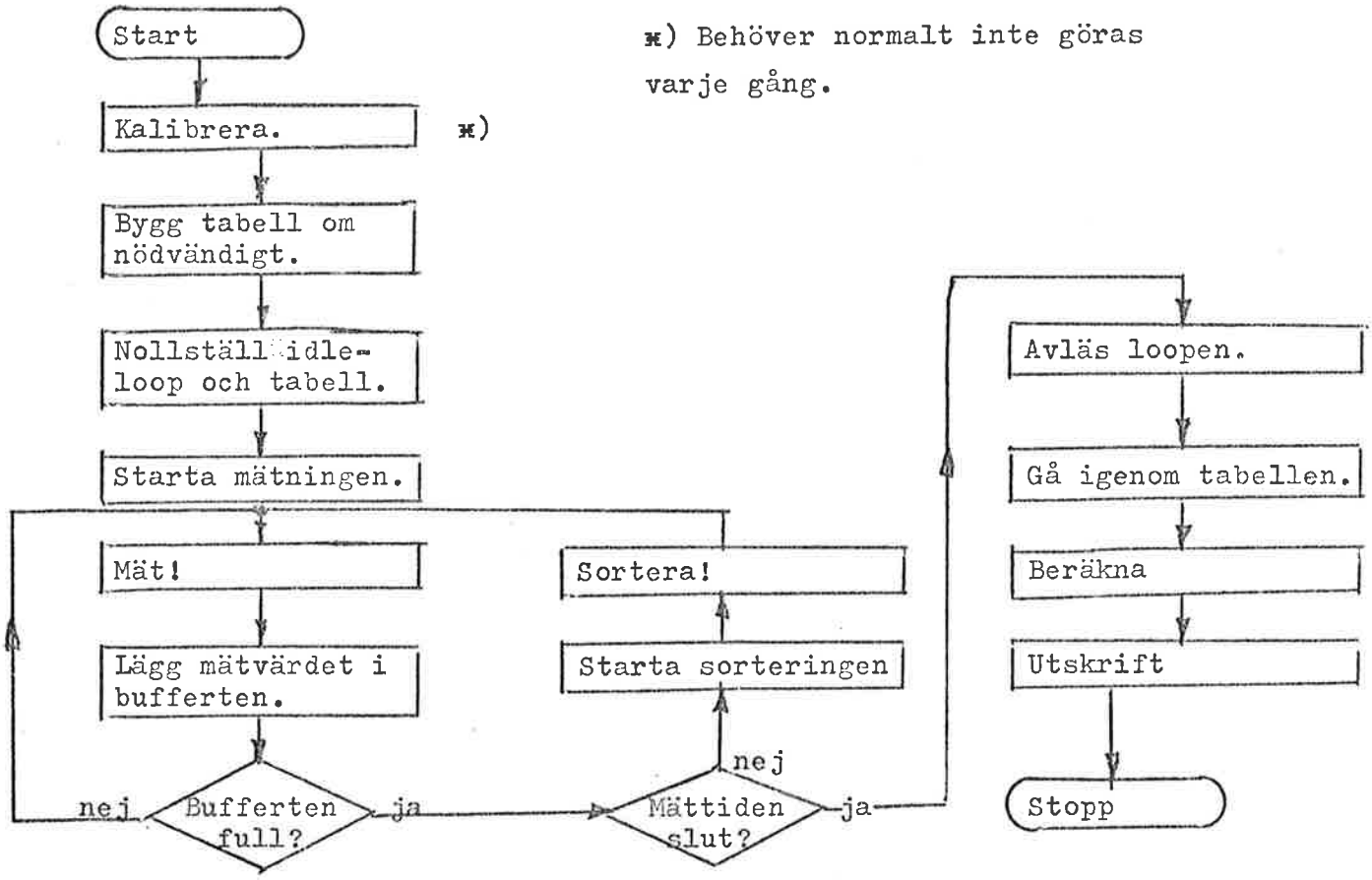
$$RA_i = \frac{RA_i^{*}}{\text{total mättid}} \qquad (4.10)$$

RA_i = i:te programmets procentuella belastning.
RA_i^{*} = i:te programmets exekveringstid.

Beräkningen enligt ovan utföres lämpligen av ett speciellt beräkningsprogram som vid mättidens slut läser av idleloopen och går igenom tabellen över mätvärdena ett efter ett.

Följden av att beräkning sker endast en gång, vid mättidens slut, är att endast medelbelastningen kan erhållas som utdata. Emellertid torde detta, då varje enskilt applikationsprograms belastning mätes, vara fullt tillräcklig information.

En sammanställning av aktiviteterna ifrån mätstart till utskrift av resultatet återfinnes i figur 4.20.



*) Behöver normalt inte göras varje gång.

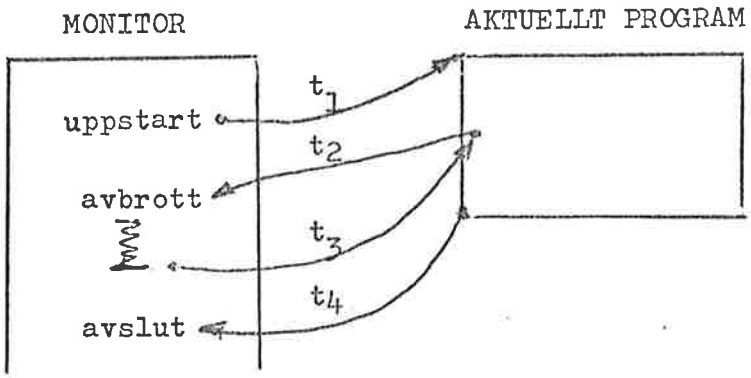
Figur 4.20 Flödesschema över de aktiviteter som skall utföras.

4.1.3.1 Tidmätning mellan uppstart och avslut av program.

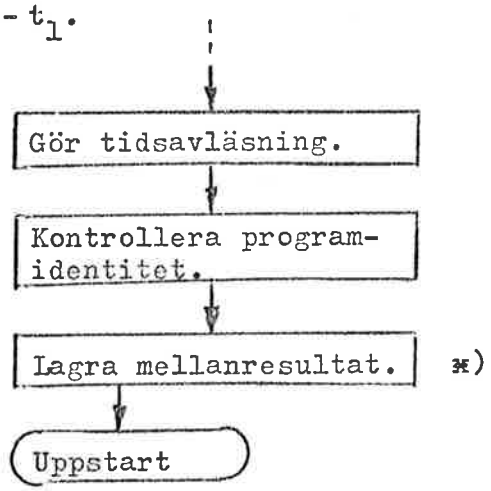
När ett program startas upp noteras starttid med hjälp av den interna klockan, samtidigt som kontroll av program-identitet sker. Vid avslut göres en ny tidsnotering och tidsskillnaden beräknas samt lagras i buffert. Sker avbrott måste tidsmätningen tillfälligt upphöra såvida inte avbrottet är direkt kopplat till programmet som exekveras. Se figur 4.21.

Då anledningen till att program lämnar ifrån sig CPU kan vara flera, exempelvis avslut och avbrott och uthopp sker till olika ställen, måste olika rutiner tillfogas.

De rutiner som krävs vid uppstart och avslut finns i figurerna 4.22 och 4.23, samt vid avbrott i figur 4.24.

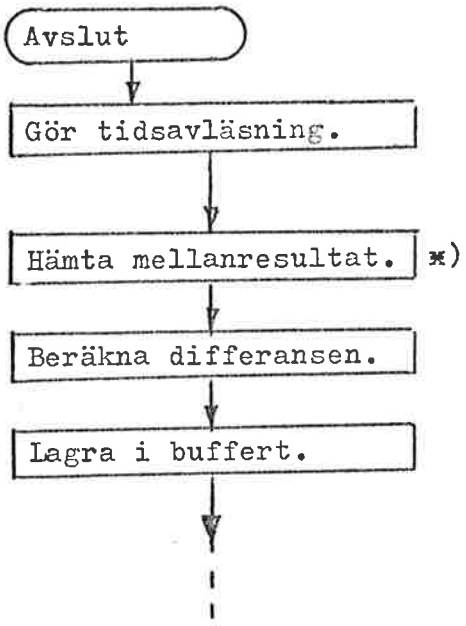


Figur 4.21 Programmets exekveringstid skall om inte avbrottet hör till det aktuella programmet beräknas som $T = (t_2 - t_1) + (t_4 - t_3)$ och inte som $T = t_4 - t_1$.

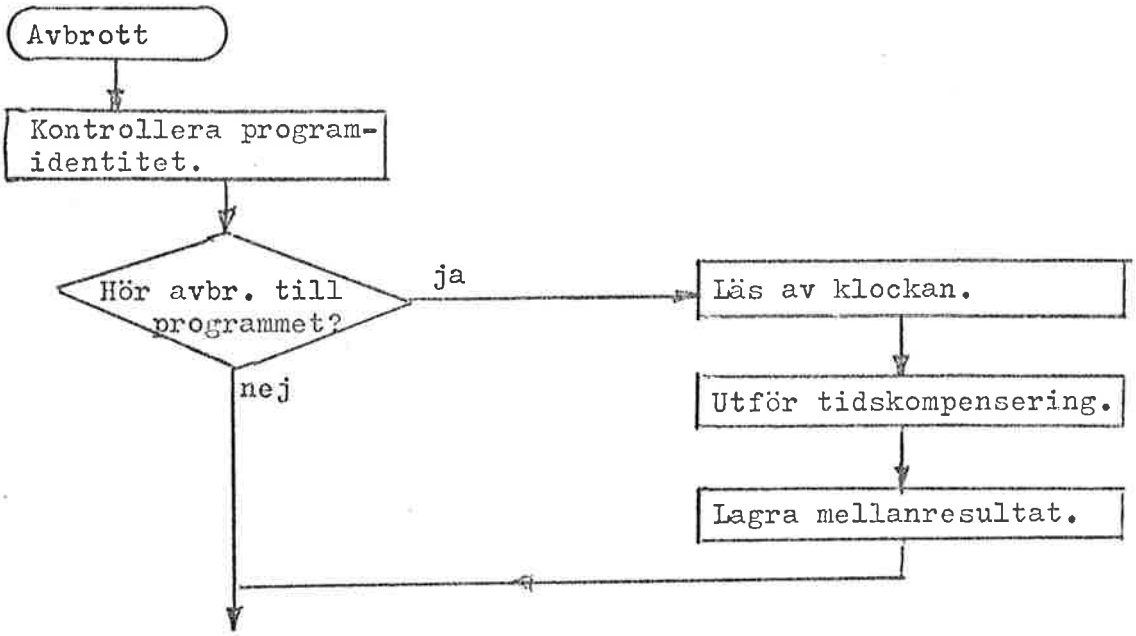


*) Detta är samma värde i de två programavsnitten. Lagringsutrymmet användes även av rutinen i figur 4.24.

Figur 4.22 Uppstartsrutin.



Figur 4.23 Avslutsrutin.



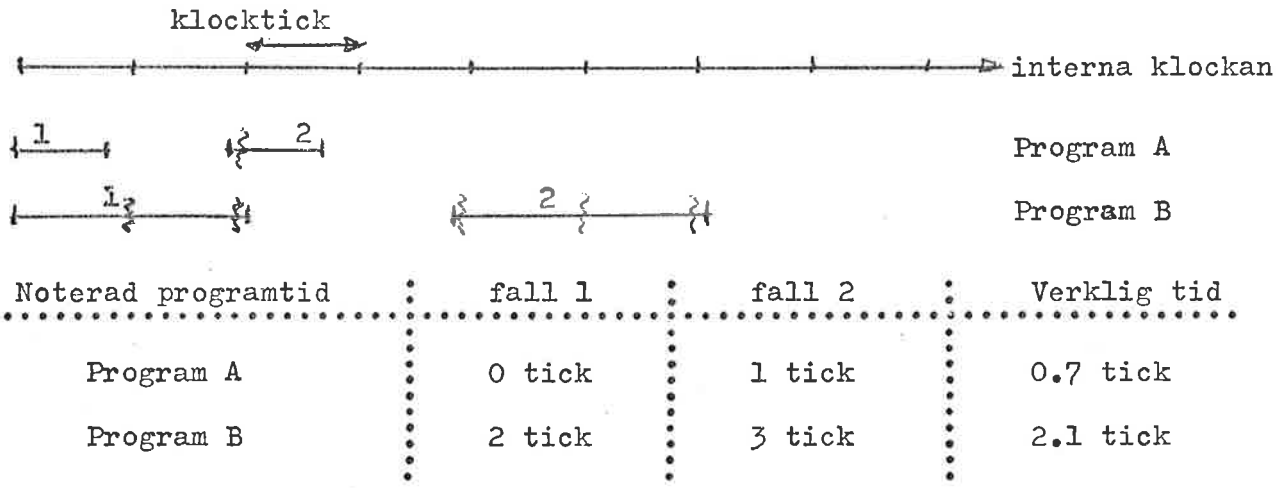
*) Kompensering kan vara nödvändig då klockan ej avläses direkt vid avbrottet.

**) Se kommentaren i figur 4.22.

Figur 4.24 Nödvändig komplettering till början av avbrottsrutinen. Tillägget i slutet kan ha samma utseende som i figur 4.23.

När den interna klockan användes kan, om upplösningen inte är tillräcklig, problem uppstå med korta program. I första hand sådana vars längd är mindre än ett klocktick. Dessa erhåller i vissa fall tidsnoteringen noll vid en exekvering. Även för program längre än ett klocktick kan fel uppstå såsom visas i figur 4.24.

Ett sätt att undvika problemet med för låg klockupplösning är att ansluta en yttre klocka som direkt kan avläsas ifrån centralenheten. En sådan klocka kan teoretiskt ges oändligt fin upplösning vilket medför att problemet enligt ovan elimineras. Anslutningen kan eventuellt, beroende på datorsystem, vålla besvär men bör ändå undersökas.



Noterad programtid	fall 1	fall 2	Verklig tid
Program A	0 tick	1 tick	0.7 tick
Program B	2 tick	3 tick	2.1 tick

Figur 4.24 Illustration till problemet med för låg klockupplösning.

Bufferten innehållande mätvärdena töms som tidigare sagts av ett sorteringsprogram (figur 4.17) med jämna mellanrum. Slutberäkning och presentation utföres av ett beräkningsprogram efer mättidens slut. Presentationen utformas lämpligen som en tabell över alla programmen med deras respektive medelbelastning.

— Positiva egenskaper hos metoden:

- Exekveringstiden mätes direkt med tidtagning, vilket ger sanna värden på belastningen, förutsatt att systematiska fel kan undvikas.

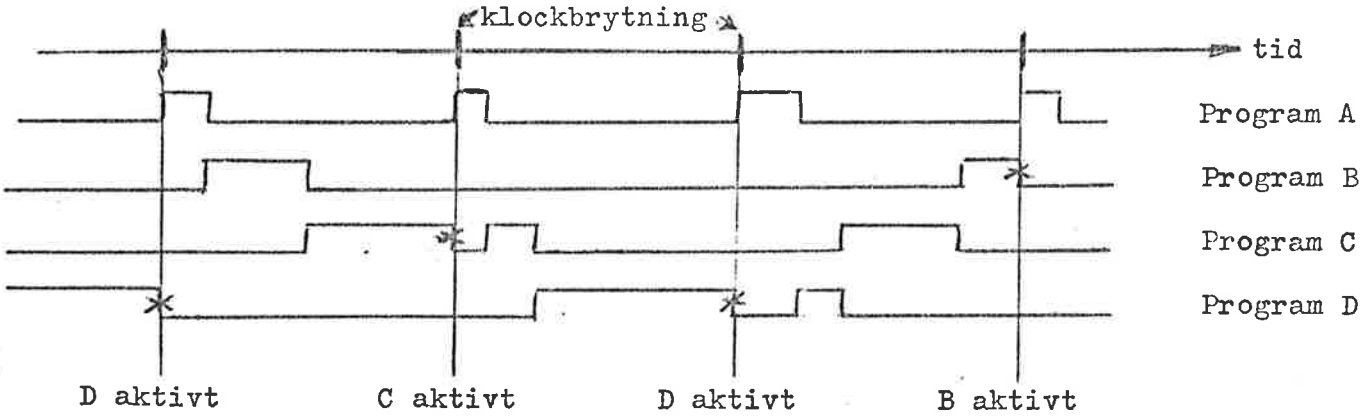
— Negativa egenskaper hos metoden:

- Mycket omfattande arbete krävs vid mätningen. Kan ge oacceptsbelt hög belastningsökning på centralenheten.
- Ingrepp och ändringar i operativsystemet nödvändiga.
- Idle-loopen är beroende av datorns cykeltid. Kalibrering är därför nödvändig.

4.1.3.2 Stickprovstagning på exekverande program vid klockbrytning.

Vid varje klockbrytning kontrolleras vilket program som exekveras. Kontrollen utföres genom ett tillägg till klockrutinen, som söker reda på progammets namn och lagrar detta i en buffert. Denna buffert töms med jämna mellenrum och i tabellen över alla programmen uppdateras en räknare för var gång respektive namn förekommer i bufferten.

Metoden lider av ett systematiskt mätfel. Program som alltid startas upp efter klockbrytning, och är kortare än intervallet mellan två på varandra följande brytningar, kommer normalt inte alls med bland mätresultatet. Detta beroende på att mätningen alltid sker vid en fix tidpunkt. Se figur 4.25.



Figur 4.25 Illustration av systematiskt fel. Program av typ A som alltid startas upp vid klockbrytning.

Det är emellertid tveksamt om detta systematiska mätfel är av så allvarlig art. Exekveringsfrekvensen för program som alltid startar upp vid klockbrytningar är nämligen normalt känd så att belastningen kan beräknas manuellt.

Eftersom mätfel trots allt uppstår bör detta tydligt påpekas vid utskriften för att undvika alla missförstånd.

Då mätresultatet är en form av stickprov, kan direkt den metodik som beskrivits på sidorna 3.11 - 3.17 användas vid slutberäkningen. Med hjälp av detta resultat och idle-loopräknarens värde kan sedan operativsystemets procentuella belastnihg erhållas ur ekvation 4.9. Intervallgränser kan beräknas som ekvation 4.11 visar.

$$RA^{*} = \sum_{i=1}^{N} (RA_i^{*} \pm \lambda_{\frac{\alpha}{2}} \cdot d_i)$$

$$RO = (\text{ekvation } 4.9) \pm \frac{\lambda_{\frac{\alpha}{2}} \cdot \sum_{i=1}^{N} d_i}{\text{total mättid} \cdot \sqrt{N}} \qquad (4.11)$$

(beteckningar enligt kapitel 3)

—— Positiva egenskaper hos metoden:

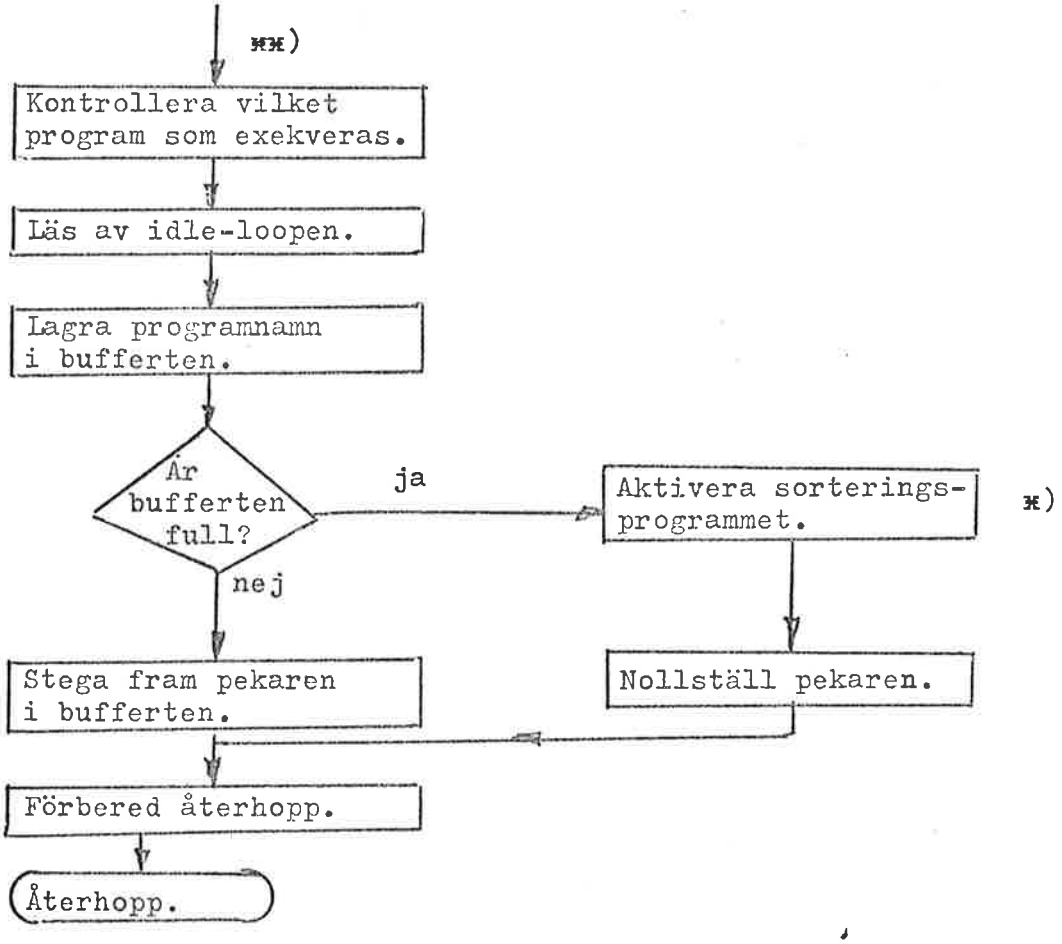
- Ingen tidtagning behövs. Därmed bortfaller problemt med klockupplösning.
- Aktiviteten i CPU behöver inte följas kontinuerligt utan endast då stickproven tas. Därmed minskar belastningen på centralenheten orsakad av mätprogrammen.

—— Negativa egenskaper hos metoden:

- Systematiska mätfel beroende på att mätning alltid sker vid klockavbrott.
- Vissa tillägg i operativsystemet är nödvändiga.
- Då metoden grundar sig på stickprovstagning erhålles alltid en viss osäkerhet i resultatet.
- Idle-loopen kräver kalibrering.

4.1.3.3 Stickprovstagning på exekverande program initierad av yttre enhet.

En yttre enhet i form av en pulsgenerator, ansluten via lämplig ingång, genererar avbrottspulser. I avbrottsrutinen utföres sedan nödvändiga åtgärder för mätning enligt figur 4.26.



*) Genom att aktivera sorteringsprogrammet via avbrottsrutinen behöver operatören inte utföra några som helst kommandon för detta.
**) Vid mätstart skall idle-loopen nollställas vilket kräver ytterligare någon instruktion.

Figur 4.26 Utformning av avbrottsrutinen.

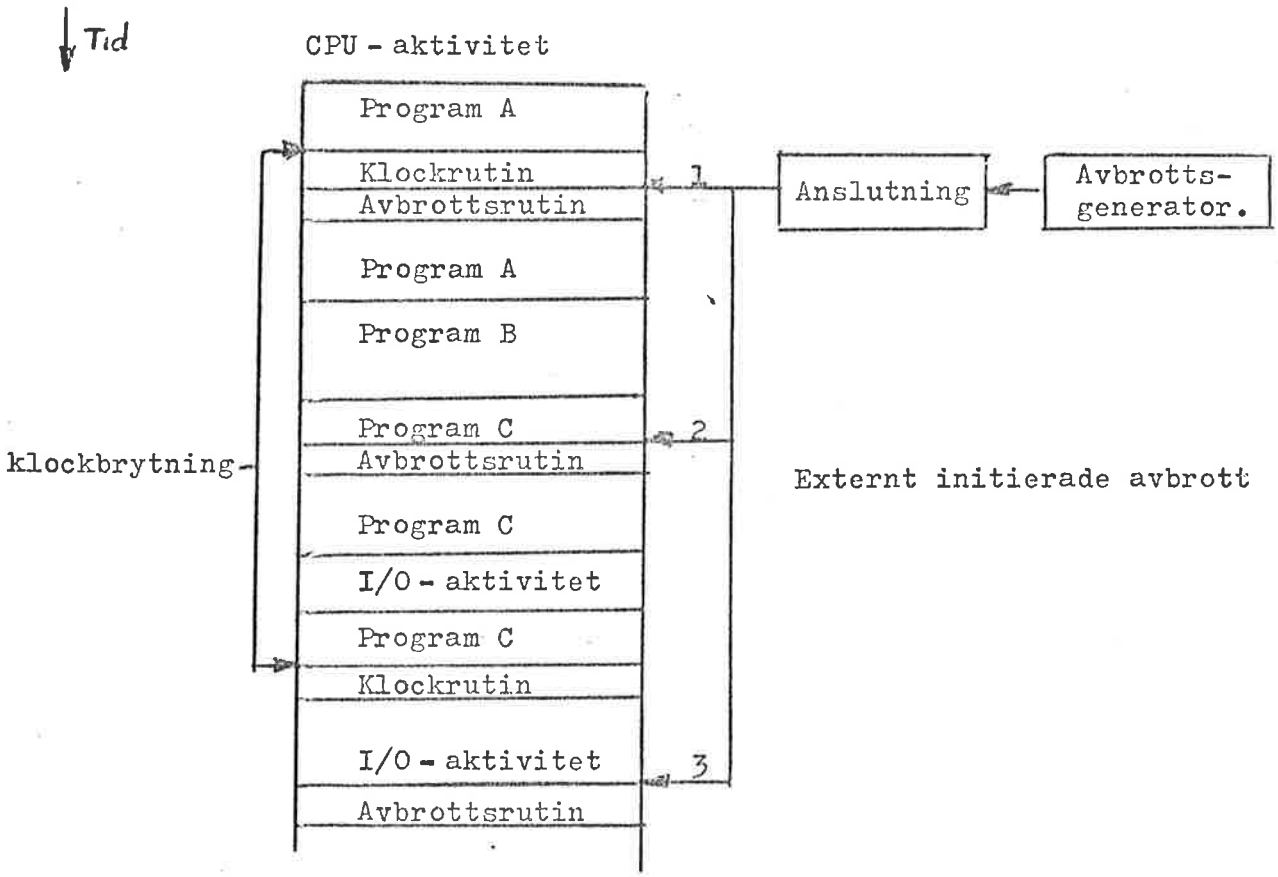
Sorteringsprogrammets uppgift är endast att uppdatera respektive räknare i tabellen över programmen, för var gång de förekommer i bufferten.

Slutberäkningarna som blir av statistisk natur utföres enligt de på sidorna 3.11 - 3.17 beskrivna metoderna. Därefter kan operativsystemets belastning beräknas enligt ekvation 4.9 och 4.11.

Avbrottsgeneratorns konstruktion är inte på något sätt kritisk. En standard pulsgenerator bör utan problem kunna användas. Det viktiga är att pulsernas amplitud och bredd

är av lämplig storlek för aktuellt datorsystem. Vidare måste pulsfrekvensen kunna justeras till ett värde som skiljer sig ifrån datorns interna klockpuls,eller multiplar därav. Detta för att undvika mätfel av den typ som visas i figur 4.25.

Om avbrottssystemet är av typen "multilevel interrupt" med möjlighet för avbrott av högre prioritet att avbryta sådana med lägre, kan även exempelvis I/O - rutinernas belastning av CPU mätas. Förutsatt att den anslutna yttre enhetens avbrottsprioritet lägges på högsta möjliga nivå. Ett exempel visas i figur 4.27.



Figur 4.27 CPU - aktivitet då mätning pågår. Vid de tre avbrotten noteras programmen A, B respektive någon I/O - aktivitet som aktiva. Observera att i avbrott 1 tvingas avbrottsrutinen vänta på klockrutinen eftersom denna har allra högsta prioritet.

En stor fördel med mätning initierad av yttre enhet är att den mycket enkelt kan startas upp utan operatörskommandon. Om pulsgeneratorn är ansluten för jämnan, påbörjas mätningen genom att sätta den i läge "on" via en vanlig strömbrytare. Mätning och sortering av mätvärden sker då ända tills avbrottspulserna upphör. Eftersom idle-loopen läses var gång enligt figur 4.26 finns alltid ett aktuellt värde på denna räknare, när avslutning av mätningen än sker.

— Positiva egenskaper hos metoden:
- Val av mätfrekvens sker helt oberoende av datorn genom ändring av frekvensen hos avbrottsgeneratorn.
- Enkelt att starta mätningen utan användning av oeratörskommandon.

— Negativa egenskaper hos metoden:
- Yttre enhet måste installeras.
- Idle-loopen kräver kalibrering.
- Eftersom mätvärdena är stickprov finns alltid en viss osäkerhet i slutresultatet.

4.1.4 Mätning av exekveringstid för varje prioritetsnivå.

4.1.4.0 Allmänt

Metodiken med särskillnad på prioritetsnivå liknar mycket den i föregående avsnitt beskrivna, särskillnad på programnivå. Isället för att notera enskilda programs exekveringstid undersökes hela prioritetsnivåers utnyttjandetid i centralenheten. Detta kan troligen göras utan att veta programmens identitet. För att operativsystemet skall kunna avgöra vilket program som först skall ges möjlighet att exekvera vid kö, måste prioriteten för varje program vara lätt tillgänglig. Detta utnyttjas vid mätningen för den nödvändiga kontrollen.

En tabell måste uppställas innan mätningen startar för lagring av resultatet. Det är då lämpligt att tilldela varje prioritetsnivå utrymme, för att vid slutbearbetningen eliminera de som inte visat sig användas. Tabellen måste även nollställas före mätningens början.

Eftersom varje prioritetsnivå normalt associeras med ett nummer, beskrivande dess prioritet i förhållande till andra nivåer, bör sortering av mätvärdena kunna lösas enkelt. Varje nivå tilldelas ett tabellutrymme, vars adress ges av en basadress plus prioritetsnummer. På så sätt kan mätvärdena snabbt placeras på rätt ställe, utan stegvis genomgång av tabell tills rätt plats hittats. Något sorteringsprogram blir heller inte nödvändigt.

Ett program som utför nödvändig beräkning och presenterar resultatet aktiveras efter mättidens slut Detta program går igenom tabellens värden ett efter ett och eliminerar ej använda prioritetsnivåer. Eftersom det i första hand är intressant att, vid denna typ av belastningsmätning, få reda på de olika nivåernas medelbelastning blir beräkningsarbetet enkelt. Betraktas operativsystemet som en separat prioritetsnivå kan belastningen i procent erhållas genom ekvation 4.12.

$$PB_i = \frac{PB_i^*}{\text{total mättid}} \qquad \text{(procent)} \qquad (4.12)$$

PB_i = i:te prioritetsnivåns belastning.

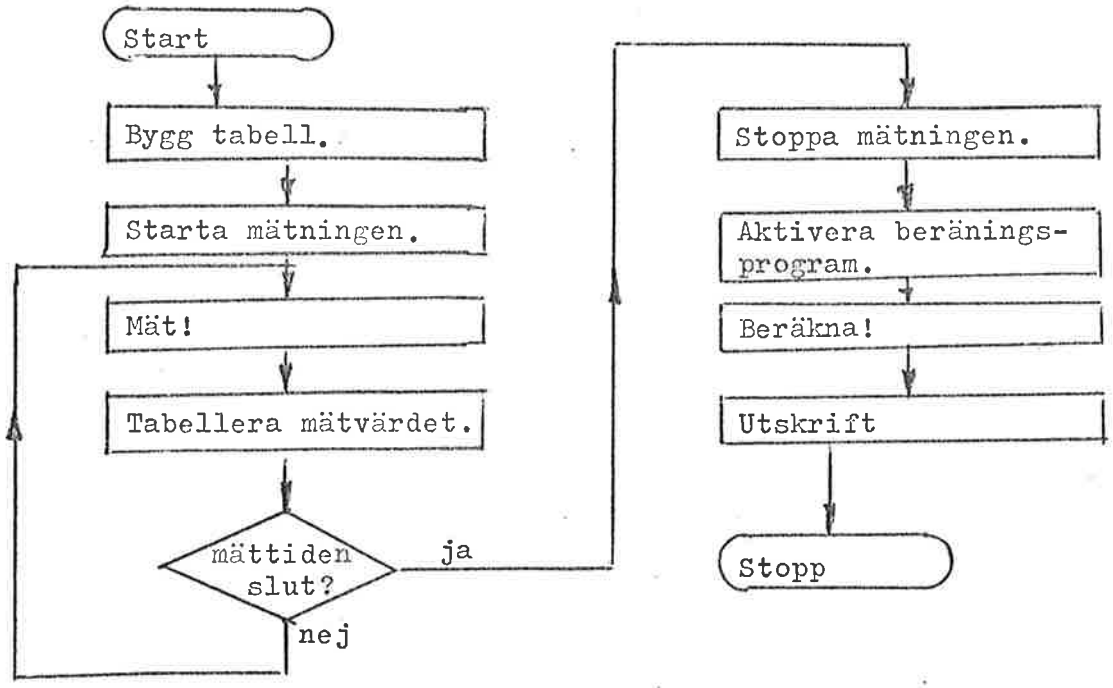
PB_i^* = i:te prioritetsnivåns uppmätta exekveringstid.

Den del då varken operativsystem eller applikationsprogram exekveras, inaktiv tid, kan uttryckas matematiskt såsom ekvation 4.13 visar.

$$IAKT = \text{total mättid} - \sum_{i=1}^{N} PB_i^{*} \qquad (4.13)$$

IAKT = Inaktiv tid.
N = antalet använda prioritetsnivåer.

En sammanställning av de aktiviteter som måste utföras från uppstart till avslut av mätproceduren, utan uppdelning på olika program finns i figur 4.28.



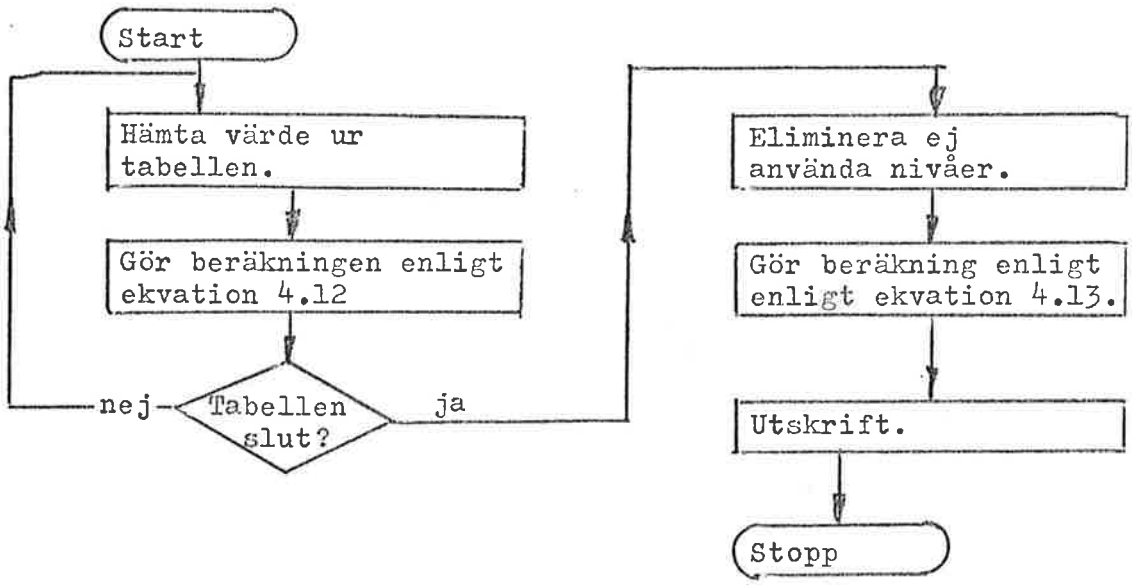
Figur 4.28 Sammanställning över de aktiviteter som skall utföras vid mätningen.

Mätningen kan i princip utföras på två sätt, genom direkt tidtagning av de olika prioritetsnivåernas exekveringstid, eller genom stickprovstagning av vilken nivå som exekveras vid mättillfället. Här kommer endast det första sättet att beskrivas.

4.1.4.1 Tidmätning med notering av prioritetsnivå.

När ett program startas upp noteras starttid och prioritetsnivå. Vid avslut sker en ny tidavläsning och använd tid beräknas och tabuleras. Sker ett avbrott som inte direkt hör till något program på den aktuella prioritetsnivån måste tidmätningen tillfälligt upphöra. Metodiken blir samma som den beskriven i "Tidmätning mellan uppstart och avslut av program" i avsnitt 4.1.3.1. Den enda skillnaden är att en särskllnad göres på prioritetsnivå istället för programnivå, samt att resultatet lagras direkt i tabell utan mellanliggande buffert. Se vidare sidorna 4.26 - 4.29.

Ett program som utför de efter avslutad mätning nödvändiga beräkningarna enligt ekvationerna 4.12 och 4.13 och presenterar resultatet kan se ut som i figur 4.29.



Figur 4.29 Utformning av beräkningsprogrammet.

Användes den interna klockan för tidavläsningen kan problem med upplösningen uppstå. Se figur 4.24 på sidan 4.29. Om datorsystemet tillåter kan då en yttre klocka eventuellt anslutas, med bättre upplösning än den interna.

—— Positiva egenskaper hos metoden:

- Exekveringstiden mätes direkt med tidtagning. Ger sanna värden på belastningen, förutsatt att systematiska fel kan undvikas.
- Enkel sortering av mätvärdena trots uppdelning i många grupper.

—— Negativa egenskaper hos metoden:

- Ingrepp och ändringar i operativsystemet ofrånkomliga.
- Eftersom mätrutinerna genomgås var gång aktiviteten i centralenheten ändras, kan rutinernas exekveringstid ge upphov till en ej försumbar belastning.

4.2 Exekveringsfrekvensen för varje applikationsprogram.

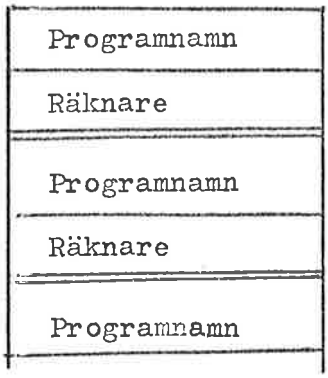
4.2.0 Allmänt

Vid mätning enligt denna princip måste, var gång ett program startas upp eller alternativt avslutas, en räknare uppdateras. Denna räknares värde lagras lämpligast i en tabell där samtliga program finns angivna.

Tabellen, som måste finnas färdig innan mätningen startar, kan byggas på minst två olika sätt. En möjlighet är att genom ett program kontrollera alla i systemet befintliga program och med dessa uppgifter sammanställa en tabell. Metodiken finns närmare beskriven i avsnitt 4.1.3.0.

Ett annat sätt är att definiera en tabell där sedan varje program manuellt tilldelas utrymme. Fördelen är att sökande i operativsystemet efter befintliga program inte är nödvändigt. Därmed förenklas mjukvaran betydligt.

De två olika varianterna av tabelluppbyggnad tillämpas i lösningsförslag 4.2.1.1 respektive 4.2.1.2. I båda fallen kan tabellens principiella utformning vara såsom figur 4.30 visar.



Figur 4.30 Principiellt utseende för tabellen för lagring av mätresulatet.

Efter mätningens avslutande genomgås tabellen av ett program som utför nödvändiga beräkningar och slutligen presenterar resultatet. Beräkningarna som behöver göras är mycket enkla, endast exekveringsfrekvensen för varje applikationsprogram behöver beräknas. Ekvation 4.14 visar hur denna erhålles ur mätvärdena.

$$EXF_i = \frac{GGR_i}{\text{total mättid}} \qquad \text{(gånger/tidsenhet)} \qquad (4.14)$$

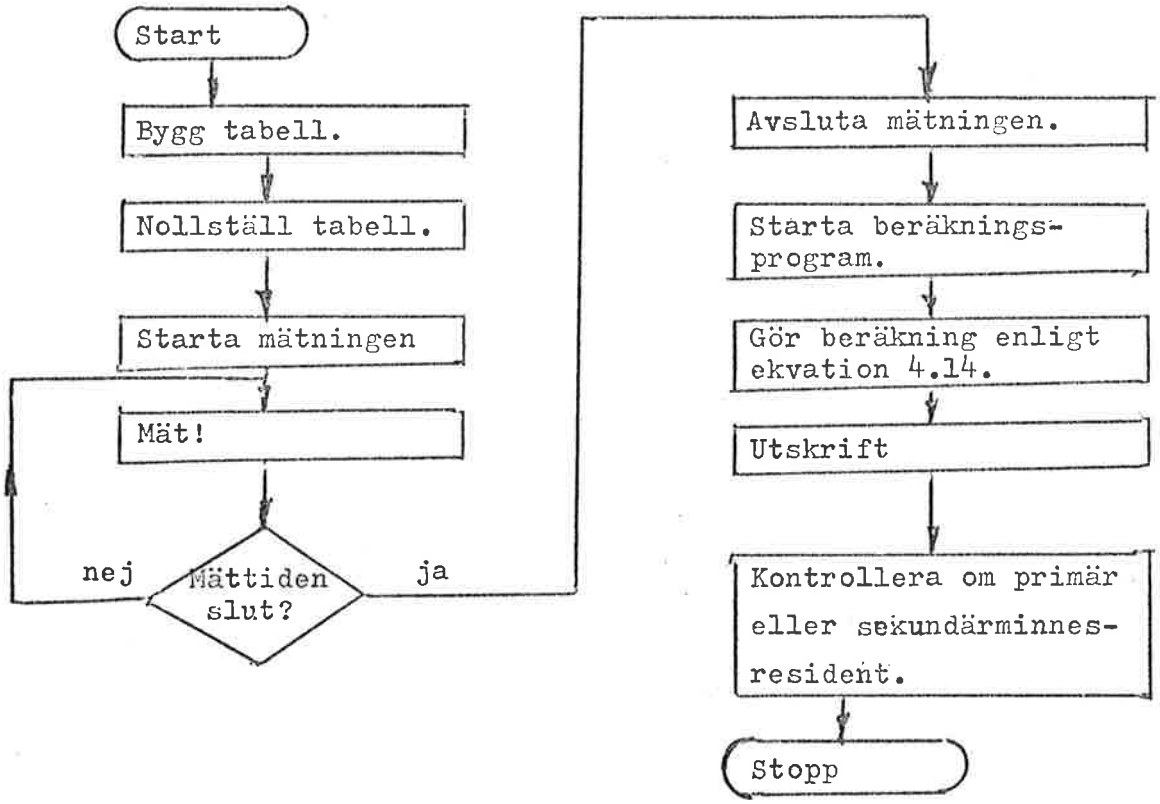
EXF_i = Exekveringsfrekvensen för det i:te programmet.

GGR_i = Antal gånger som program i startats upp eller avslutats, d.v.s. räknarvärdet i tabellen.

Vid en mätning av exekveringsfrekvensen är det lämpligt att notera vilka program som är primär respektive sekundärminnesresidenta. För att inte komplicera mätrutinerna mer än nödvändigt sker detta enklast manuellt.

Anledningen till kontrollen är, som nämndes i kapitel 3, att med kännedom om exekveringsfrekvensen kan en mer optimal indelning av programmen mellan olika minnestyper ske. Program med hög exekveringsfrekvens bör om möjligt placeras i primärminnet för att undvika onödig minnesöverföring.

En sammmanställning över de aktiviteter som måste utföras för mätningens genomförande finns i figur 4.31.

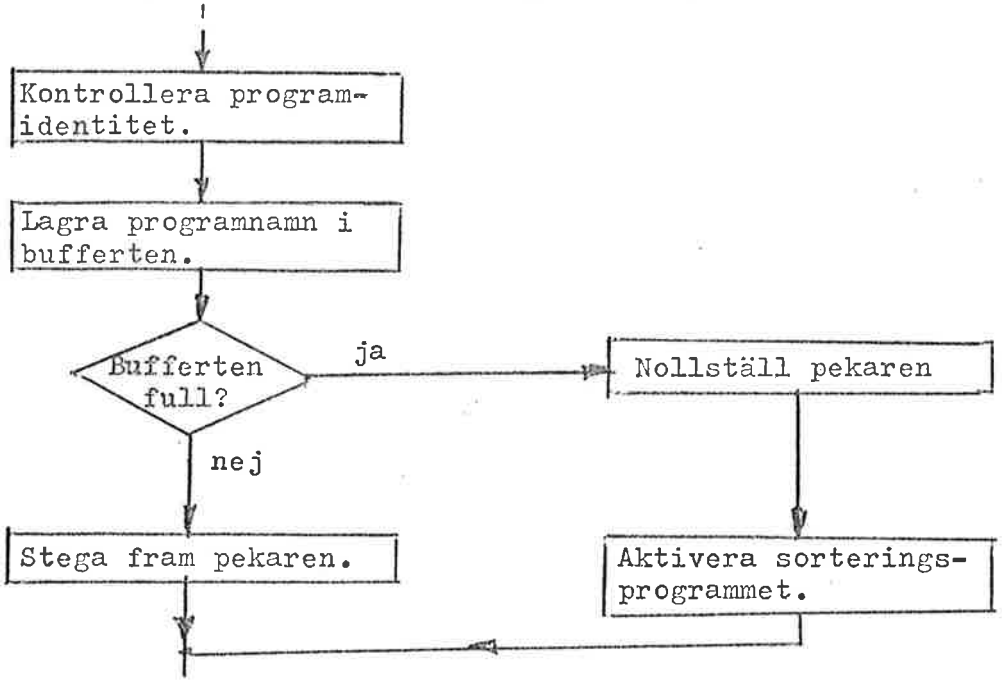


Figur 4.31 Sammanställning över de aktiviteter som skall utföras vid mätning.

4.2.1.1 Modifiering av STOP - rutinen.

I STOP - rutinen tillfogas ett avsnitt som vid avslut av ett program kontrollerar dess identitet och uppdaterar en räknare, Räknaren finns i en , innan mätningen mätningen påbörjats, uppställd tabell över samtliga program.

Eftersom sökandet efter rätt tabellplats kan ta lång tid om många program finns i systemet, är det lämpligt att införa en buffert för lagring av mellanresultat. Bufferten töms sedan, när den är full, av ett sorteringsprogram som uppdaterar tabellen. Utseendet av avsnittet som skall tillfogas STOP - rutinen framgår av figur 4.32.



Figur 4.32 Tillägget till STOP - rutinen.

Eftersom mätningen styrs direkt genom STOP - rutinen är det inte lämpligt , knappast ej heller möjligt, att enkelt starta och stoppa den via operatörskommando. Därfär bör mätstart och mätstopp styras via ett annat program. Detta program aktiveras när mätningen önskas starta och bygger då en tabell samt nollställer denna. Reaktivering sker sedan när mättiden är slut och programmet kopierar då den fyllda

tabellen för att starta beräkningsarbetet. Anledningen till kopieringen är att mätrutinen hela tiden, om beräkningsprogrammet blir avbrutet av andra program med högre prioritet, fortsätter att leverera mätvärden.

—— Positiva egenskaper hos metoden:

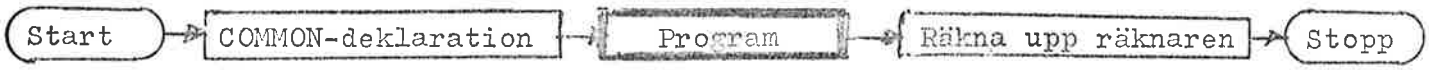
- Ingen tidtagning behövs med därav följande risker för fel orsakade av dålig klockupplösning eller bristande kalibrering.
- Enkla och entydiga mätresultat.

—— Negativa egenskaper hos metoden:

- Ingrepp i operativsystemet nödvändiga.

4.2.1.2 Modifiering av varje program.

Varje program kompletteras med ett avsnitt som uppdaterar en räknare i en COMMON-area vid varje exekvering. Plats i arean tilldelas manuellt genom att direkt i det tillfogade avsnittet explicit ange lagringsplats för mätvärdena. Följden blir att tillägget till varje program får något olika utseende. Skillnaden inskränker sig dock till en adressangivelse. Utseendet på tillägget framgår av figur 4.33.



Figur 4.33 Tillägg till varje program.

COMMON-arean är tillgänglig för ett program som innan mätningen startar, nollställer alla värden och efter mättidens slut kopierar samtliga värden innan slutberäkning. Anledningen till kopieringen är samma som i föregående metod, d.v.s. att uppdatering av räknarna sker kontinuerligt, genom det i varje program inlagda avsnittet.

En mätmetod enligt denna princip lämpar sig bäst för inpassning i nya system, där erfordeliga kompletteringar enkelt kan ske innan olika program läses in. Sker inläsningen via hålkort behöver endast några nya kort tillfogas till varje program. I redan befintliga system blir det förberedande arbetet betydligt mer omfattande.

—— Positiva egenskaper hos metoden:
- Ingen tidtagning behövs med därav följande risker för fel orsakade av dålig klockupplösning eller bristande kalibrering.
- Enkla programavsnitt att tillfoga.
- Enkla och entydiga mätresultat.

—— Negativa egenskaper hos metoden:
- Omständigt att till varje program göra tillägg.
- Manuell uppbyggnad av tabell ger extraarbete.
- De gjorda tilläggen kan inte enkelt tas bort när mätning inte längre är aktuell.

4.3 Primärminnesbeläggning.

4.3.0 Allmänt

Vid mätning av promärminnesbeläggning kommer, som påpekets i kapitel 3, en inskränkning till minnen utan indelning i partitioner att göras. Orsaken är att mätning i första hand är intressant när minnet består av en enda större sektor där alla program skall samsas om utrymme. När partitionsindelning förekommer blir minnestilldelningen styrd på ett helt annat sätt och en mätning får betydligt mindre värde.

Två olika principer kommer att behandlas. Det ena, som kanske faller sig mest naturligt, är att räkna antalet upptagna, alternativt lediga ord. På så sätt fås ett exakt värde på beläggningstillståndet. Ett alternativt förfarande är att istället kontrollera vilka program som vid

mättillfället befinner sig i minnet. Med kännedom om deras fusiska längd kan sedan totalt minnesutnyttjande beräknas.

Mätningen kan endera ske kontinuerligt, så att varje ändring noteras, eller som stickprov med vissa mellanrum. Eftersom en kontinuerlig registrering blir komplicerad och tidskrävande bör en stickprovstagning användas. Styrningen av mätningen sker då antingen via operativsystemet så att ett mätprogram aktiveras med jämna mellanrum eller med en yttre enhet som genererar avbrott. I avbrotten sker då den nödvändiga kontrollen av minnesbeläggningen. En styrning via operativsystemet är tveklöst enklast att använda och de följande lösningsförslagen baserar sig på denna variant.

Ett lämpligt sätt att presentera resultatet är det på sidan 3.6 och följande beskrivna med belastningens fördelning. Eventuellt kompletterad med registrering av konsekutivt lika beläggning.

För att utföra nödvändigt sorterings och beräkningsarbete åtgår ganska mycket tid. Det är därför lämpligt att lagra mätvärdena i en buffert vid mättillfället. Bufferten töms sedan av ett sorteringsprogram som sorterar och beräknar enligt tidigare skisserade metoder.

Då behandlingen av mätvärdena skiljer sig åt i de följande två lösningsförslagen redovisas metodiken under respektive avsnitt.

4.3.1.1 Direkt räkning av upptaget utrymme.

En rutin undersöker det område i operativsystemet där uppgifter om primärminnesbeläggningen finns lagrad. Det som därvid skall kontrolleras är de olika programmens disponerade utrymme. Eftersom det knappast är intressant att veta exakt v i l k e t område som är belagt, utan endast hur mycket, kan de olika programmens utnyttjade område summeras direkt. Den procentuella minnesbeläggningen kan därefter enkelt erhållas som utnyttjat område i förhållande till totalt tillgängligt.

Några generella aspekter på hur nödvändiga uppgifter kan framtagas är svårt att ge eftersom metodiken helt beror på operativsystemets uppbyggnad. Eventuellt kan problem med minnesskydd uppstå, men dessa bör i de flesta fall relativt enkelt kunna lösas.

—— Positiva egenskaper hos metoden:
- Enkla och lättolkade mätvärden.
- En direkt information om minnesbeläggningen erhålles.

—— Negativa egenskaper hos metoden:
- Tillgång till av operativsystemet använda minnesareor nödvändig. Därav blir metodens konkreta utformning helt beroende av aktuellt system.
- Mätningen tar troligen ganska lång tid vid varje mättillfälle, då hela primärminnets beläggning måste kontrolleras.

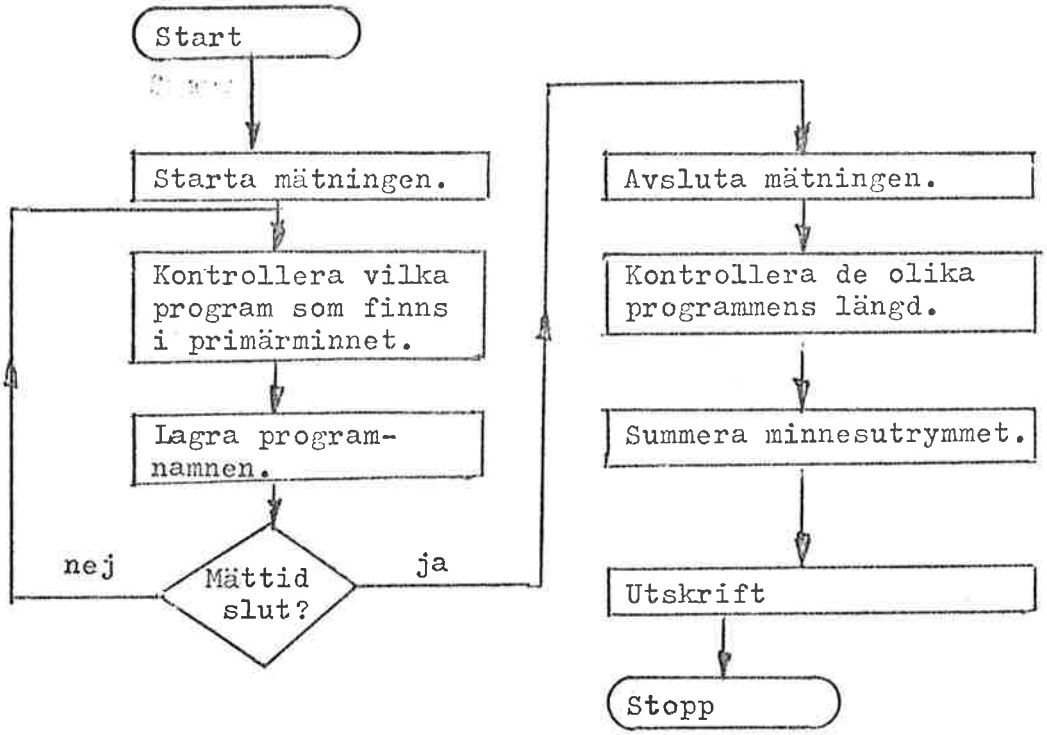
4.3.1.2 Kontroll av i primärminnet befintliga programs längd.

En rutin undersöker vid mättillfället vika program som finns i minnet. Genom kontroll av varje programs längd kan sedan totala utnyttjandegraden beräknas.

Kontrollen av programlängden kan endera ske manuellt eller programmässigt. Vid manuell kontroll uppskattas varje programs längd och summering av de vid mättillfället i minnet befintliga programmen utföres. Användes programmässig kontroll kan en metodik som skisserats i figur 4.34 användas. Möjlighet att erhålla information om programmens minnesbehov varierar naturligtvis med olika system.

—— Positiva egenskaper hos metoden:
- Enkla och lättolkade mätvärden.
- En direkt information om minnesbeläggningen erhålles.

—— Negativa egenskaper hos metoden:
- Tillgång till av operativsystemet använda arbetsareor nödvändig, vilket medför starkt systemberoende.
- Kräver ganska omfattande arbete efter mätningens slut med kontroll av programlängder.



Figur 4.34 Nödvändiga åtgärder vid genomförandet av mätningen.

4.4 Kölängd

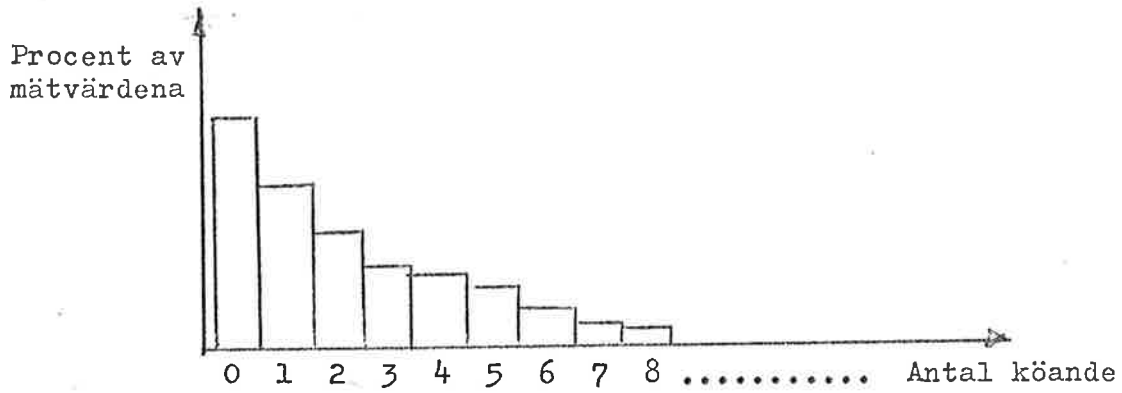
4.4.0 Allmänt

Mätning av kölängd är ett något annorlunda sätt att få reda på belastningen än tidigare redovisade metoder. Trots att mätmetoden inte ger ett direkt mått på belastningens storlek torde den ändå vara attraktiv genom sin enkelhet. Endera kan antalet program som köar direkt räknas eller också kontrolleras enbart om kö finns eller ej.

Att undersöka om kö finns eller ej blir naturligtvis enklare än att räkna antalet köande program. Den mindre mängden information blir ej heller någon väsentlig nackdel. Avgörande i de flesta fall torde nämligen vara om ett program tvingas vänta eller ej, inte hur många som eventuellt finns före i kön.

Kö kan uppstå på flera ställen i ett datorsystem. Då mätmetodiken i olika fall emellertid blir i stort sätt likadan, endast minnesutrymmet där uppgifterna skall hämtas varierar, sker ingen särskillnad vid den fortsatta behandlingen.

Efterbehandlingen av resultatet blir enkel. Om antalet köande har räknats sker presentationen av mätvärdena lämpligen som en uppställning av fördelningen för kölängden enligt figur 4.35 Se även avsnitt 3.0.1 C.



Figur 4.35 Lämpligt presentationssätt vid räkning av antal köande.

Noteras endast om kö finns eller inte, bör det vara fullt tillräckligt att ange i hur stor del av mätpunkterna som kö fanns. Då angivet som ett procentvärde.

4.4.1.1 Kontroll om något köande program finns.

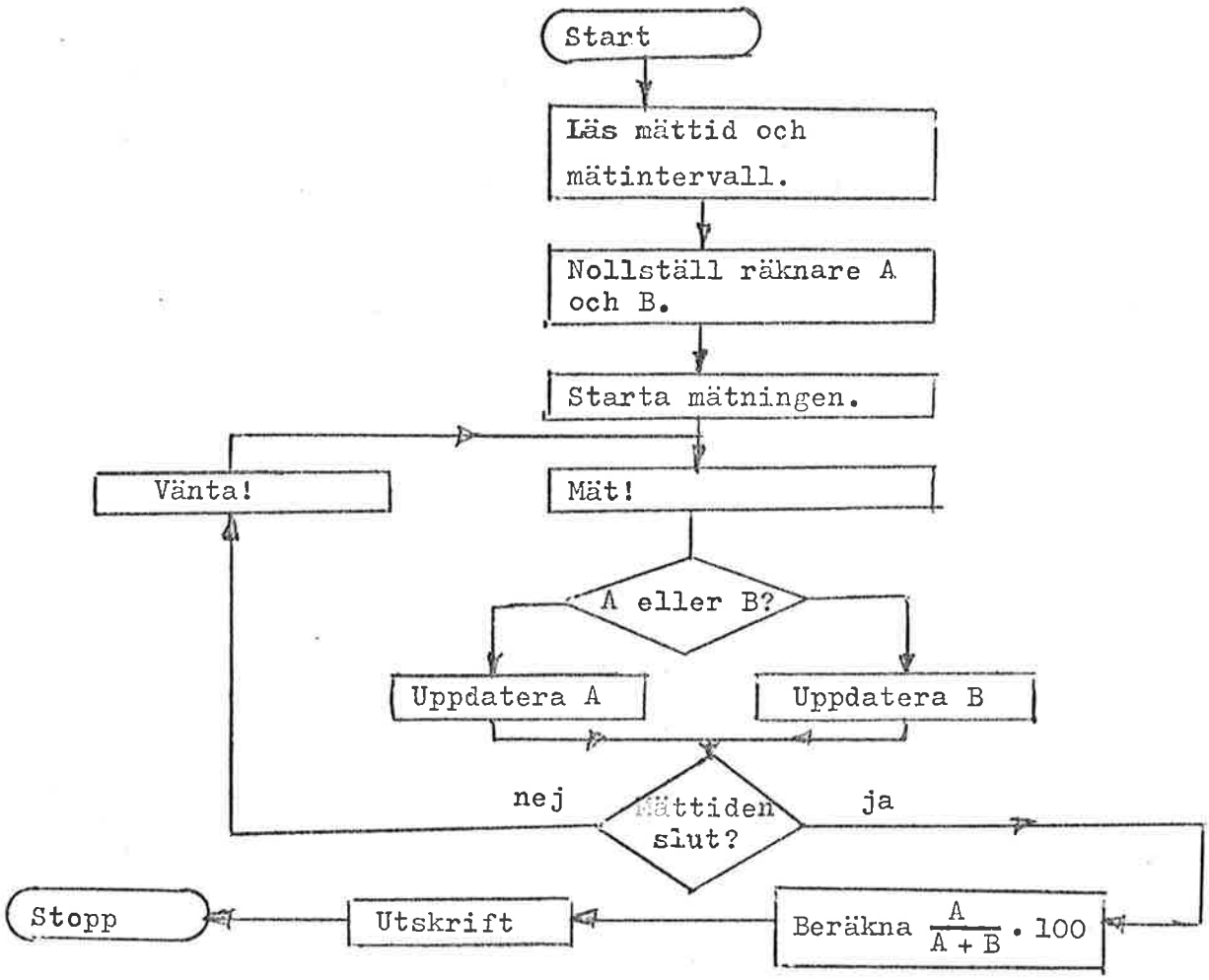
En rutin går in i den area i operativsystemet där uppgifter om kö finns och avläser huruvida något program för tillfället köar. Om så är fallet uppdateras en räknare A, annars uppdateras en annan räknare B. Totala antal mätningar blir då A + B och procentuella andelen mättillfällen då kö funnits beräknas enkelt enligt ekvation 4.15.

$$\text{Köfördelning} = \frac{A}{A+B} \cdot 100 \quad \text{(procent)} \qquad (4.15)$$

Tiden mellan mätpunkterna styrs enklast mjukvarumässigt och göres variabel via operatörskommando. Den totala mättiden skall väljas så att tillräcklig noggrannhet erhålles i resultatet. Därvid kan en intervallskattning enligt ekvation 3.1 på sidan 3.12 användas. För att resultatet skall bli representativt för verkliga förhållanden måste emellertid kontrolleras, att den tid då mätning utföres verkligen är representativ för den belastningssituation man önskar mäta.

En sammanställning över de aktiviteter som skall utföras vid mätning finns i figur 4.36.

Exakt hur mätningen skall tillgå blir helt beroende på operativsystemets uppbyggnad och några generella synpunkter är svårt att ge.



Figur 4.36 Mätning om kö finns eller ej.

—— Positiva egenskaper hos metoden:
- Enkla och lättolkade mätvärden.
- Mätproceduren är okomplicerad och kan utföras snabbt.

—— Negativa egenskaper hos metoden:
- Tillgång till av operativsystemet använda minnesareor nödvändig.
- Ger ingen direkt infrmation om belastningen och dess fördelning.

4.4.2.1 Mätning av antalet köande program.

En rutin räknar det antal program som vid mättillfället finns i den aktuella kön. Antalet lagras i en buffert för senare bearbetning. Detta för att mätproceduren i så liten omfattning som möjligt skall belasta systemet och därvid ge upphov till systematiska fel.

Om olika köer användes för olika prioritetsnivåer måste samtliga dessa köer summeras, med eventuell notering om varje kös längd innan dess.

Bufferten töms lämpligen av ett spciellt program som aktiveras när bufferten är fylld. Den bearbetning som utföres bör vara beräkning av kölängdens fördelning enligt figur 4.35. Dessutom kompletterat med ett medelvärde för hela mättiden.

Mätningen styrs enklast mjukvarumässigt med aktivering av mätrutinen genom operatörskommando. Mätrutinen startar sedan det program som tömmer bufferten.

—— Positva egenskaper hos metoden:
- Enkla och lättolkade mätvärden.
- Ger större mängd information än den föregående metoden med enbart notering om kö eller inte kö.

—— Negativa egenskaper hos metoden:

- Tillgång till av operativsystemet använda minnesareor nödvändig. Medför systemberoende lösning.
- Ger ingen direkt information om belastningen och dess fördelning.

4.5 Utnyttjandegraden av I/O - kanaler.

4.5.0 Allmänt

När utnyttjandegraden hos en kanal skall mätas, är det endast om kanalen är upptagen eller inte, som är intressant storhet. Således är det inte nödvändigt att undersöka vilket program som utnyttjar aktuell kanal. Detta förenklar mätmetodiken betydligt och två skilda principer principer kommer att beskrivas. Den ena med utnyttjande av mjukvara och den andra med hjälp av hårdvara i form av yttre ansluten räknare.

Användes yttre hårdvara kan mätning ske kontinuerligt helt oberoende av datorns inre tidbas. När softvara i form av programrutiner sköter mätningen kan kontinuerlig registrering knappast användas. Detta beroende på att all tidtagning av upptagettillstånd blir beroende av datorns inre tidsupplösning. Denna är i de flesta fall ej tillräckligt noggrann. Istället bör en stickprovstagning, av tillståndet hos kanalen, med jämna mellanrum användas.

Presentationen av mätvärdena bör ske som ett medelvärde av den del av totalt förfluten mättid som kanalen var upptagen. Vid stickprovstagning bör detta kompletteras med angivande av noggrannhet enligt 3.1 sidan 3.12.

4.5.1.1 Kontroll i operativsystemet av I/O - aktivitet.

En rutin kontrollerar om aktuell kanal noterats som upptagen eller inte. Kontrollen måste troligtvis göras på bitnivå i operativsystemet. Beroende på om upptaget-tillstånd råder eller inte uppdateras en räknare A eller B. Den del av totaltiden som kanalen varit upptagen, kan sedan mätningen avslutats, enkelt erhållas ur ekvation 4.16.

$$\text{Utnyttjandegraden} = \frac{A}{A+B} \cdot 100 \quad \text{(procent)} \qquad (4.16)$$

Rutinen aktiveras lämpligast genom operativsystemet med jämna mellanrum. Dessa mellanrum bör, liksom totala mätiden, enkelt kunna varieras via operatörskommandon innan mätstart. En varningär dock befogad vid mätning med konstanta tidsintervall. Genom att denna alltid har en period som är en multipel av den interna klockfrekvensen kan eventuellt systematiska fel uppstå. Andra aktiviteter, som också är en multipel av klockfrekvensen och använder sig av den kanal på vilken mätning sker, kan i vissa fall helt falla bort ur mätresultatet.

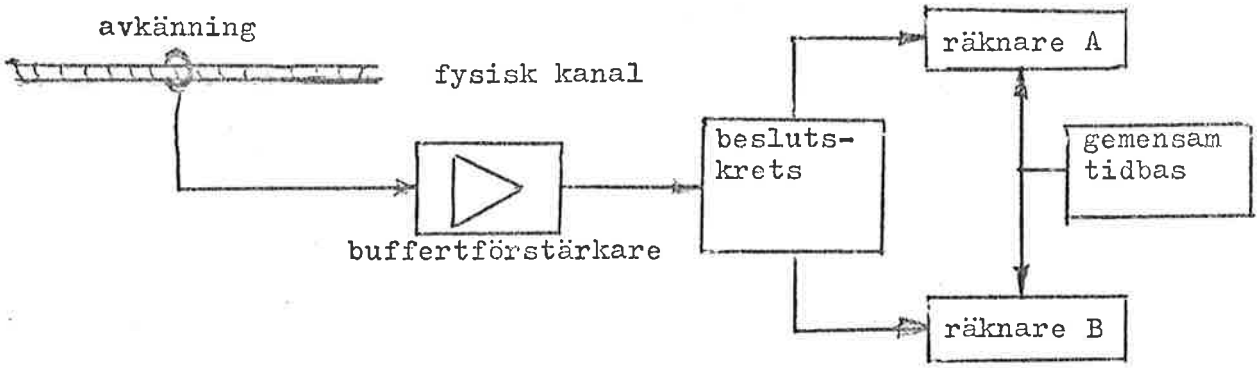
De nödvändiga aktiviteterna som måste företas vid mätproceduren följer de som finns skisserade i figur 4.36 på sidan 4.48.

— Positiva egenskaper hos metoden:
- Ett fåtal mätstorheter att särskilja. Därav enkel efterbehandling av resultatet.
- Möjlighet att mäta på flera kanaler i samma mätomgång.

— Negativa egenskaper hos metoden:
- Kräver kännedom om operativsystemet. Därav systemberoende lösning.
- Risk för systematiska fel.

4.5.1.2 Mätning av I/O - aktivitet med hjälp av en yttre enhet.

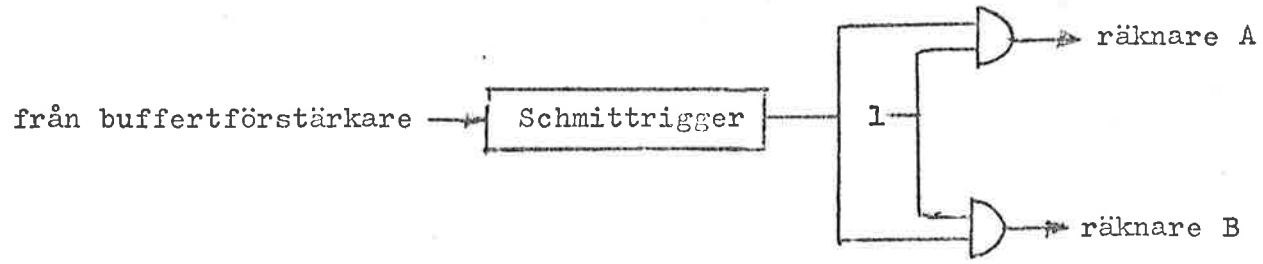
En yttre enhet kopplad direkt på den fysiska kanalen kontrollerar om denna användes eller ej. Enheten känner via en buffertförstärkare av om någon signalöverföring sker. Är så fallet räknar en räknare kontinuerligt med en ifrån datorn helt oberoende tidbas. När ingen signalöverföring sker räknar en annan räknare så att total mättid kan erhållas ur de båda räknarnas sammanlagda värden. Se figur 4.37.



Figur 4.37 Inkoppling av mätutrustningen. Beslutskretsen avgör om räknare A eller B skall ges räkneimpuls.

Avkänningen kan utformas på två sätt. Det ena är att kontrollera logisk nivå på den ledning i kanalen som styr överföringen. Denna logiska nivå användes sadan direkt för styrning av de båda räknarna. En annan möjlighet är att känna av om några strömmar flyter genom kanalen. Är så fallet pågår överföring av information och den därmed associerade räknaren skall ges impuls för räkning.

Buffertförstärkaren kan vara en enkel förstärkare med hög inimpedans och tillräckligt hög utsignal för att styra beslutskretsen. Denna krets kan se ut som i figur 4.38.



Figur 4.38 Beslutskretsens utformning. Schmittriggern anpassar signalnivån till OCH-grindarna.

Som räknare användes lämpligast standard frekvensräknare med gemensam tidbas. Fördelen med gemensam tidbas är att denna inte behöver vara känd utan enbart någorlunda konstant i tiden. Utnyttjandegraden kan då beräknas som ekvation 4.17.

$$\text{Utnyttjandegrad} = \frac{\text{räknare A:s värde}}{\text{A + B :s värde}} \qquad (4.17)$$

— Positiva egenskaper hos metoden:
- Ger ingen som helst belastning.
- Enkelt att anpassa till olika system.
- Trivial efterbehandling av resultatet.

— Negativa egenskaper hos metoden:
- Kräver anslutning av yttre hårdvara.

K A P I T E L 5

SYNPUNKTER PÅ PRAKTISKT ANVÄNDANDE

5 Synpunkter på praktiskt användande.

Några generella synpunkter på det praktiska användandet kommer här att ges. De olika metoderna att mäta belastning på som skisserats i kapitel 4, har visserligen mycket olika behov ifråga om faciliteter i form av mjuk och hårdvara, men vissa grundprinciper kan formuleras.

Ett antal punkter som steg för steg talar om vad som måste utföras vid en mätning har uppställts. Efter uppställningen följer kommentarer till var och en av dessa.

5.1 Ladda mätprogram.
5.2 Anslut eventuell yttre enhet.
5.3 Ladda avbrottsrutin för yttre enhet.
5.4 Ge parametervärden.
5.5 Starta mätningen.
5.6 Kontrollera att mätningen förlöper normalt.
5.7 Avsluta mätningen.
5.8 Ladda och aktivera program som utför slutberäkning och presentation av mätresultatet
5.9 Analysera resultatutskrift.

Samtliga punkter kan inte tillämpas på alla skisserade mätmetoder utan i vissa fall användes endast ett fåtal.

5.1 Ladda mätprogram

Samtliga de redovisade metoderna utom en kräver någon form av mätprogram. Den som inte använder sig av sådant är "mätning av I/O - aktivitet med hjälp av yttre enhet". I de övriga fallen bör det aktuella programmet finnas befintligt på massminne som genom som genom ett operatörskommando kan kallas in till primärminnet, där det finns under hela mätningen.

I de fall då tillgång till priviligierade minnesareor är nödvändig måste mätprogram ges möjlighet att operera i dessa areor. Dessutom måste prioriteten väljas så att mätningen verkligen utföres vid de önskade tidpunkterna.

När modifieringar i operativsystemet behöver införas är det lämpligt att en speciell version av detta finns lagrad på massminne. Denna modifierade version laddas sedan vid varje mättillfälle in i aktuell dator, där den tillfälligt ersätter ordinarie operativsystem. Fördelen med ett sådant förfarande är att mätrutinerna inte behöver belasta systemet under ordinarie drift. Vidare undvikes risken för att ändringarna medfört något fel som visar sig först vid normal användning utan mätning.

5.2 Anslut eventuell yttre enhet.

Även här skiljer sig metoden "mätning av I/O - aktivitet med hjälp av yttre enhet" ifrån de övriga. Där skall en räknare inkopplas på lämpligt ställe direkt på någon ledning eller kopplingsplint. I övriga fall där yttre enhet användes består denna av en pulsgenerator som kopplas till en extern avbrottsingång. Via något kommando ifrån operatören måste sedan anges att anslutning skett. Generatorn ställs därefter in på lämpliga värden vad det gäller pulsamplitud och pulslängd.

5.3 Ladda avbrottsrutin för yttre enhet.

En avbrottsrutin måste införas i de fall pulsgenerator anslutits. I rutinen utföres endera direkt de nödvändiga mätningarna eller också sker uthopp till ett speciellt mätprogram.

Om en speciell ingång alltid reserveras för denna typ av anslutning kan avbrottsrutinen finnas på plats för jämnan. Den yttre enheten kan då direkt anslutas och starta mätförloppet.

5.4 Ge parametervärden

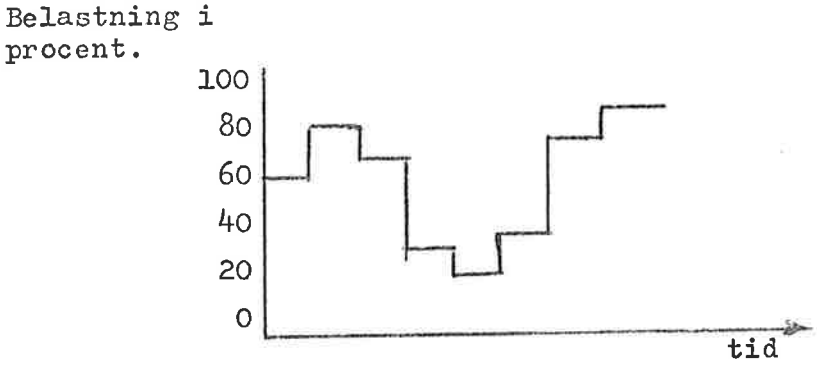
I vissa fall förekommer att en eller flera parametrar måste specificeras innan mätningen startar. Parametrar som direkt skall användas av mätprogrammet. Förekommande parametrar är i första hand total mättid, avstånd mellan varje mätning och noggrannhetsangivelser.

Totala mättiden bör, om den användes i mätprogrammet, kunna uttryckas i minuter för att få bekväm storlek på använda tal. Val av lämplig tid är starkt beroende av flera saker. En kort mättid ger sämre förutsägelser om belastningens genomsnittliga beteende än en längre. Däremot ger den bättre upplysning då ett i tiden begränsat beteende vill studeras, exempelvis vid en kortvarig toppbelastning.

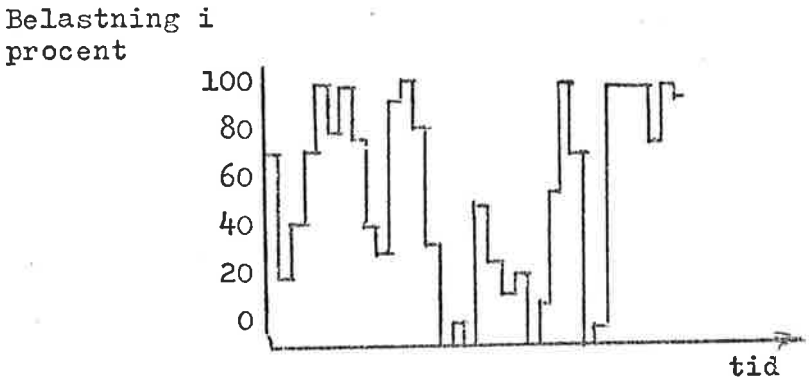
Även avståndet mellan mätningarna är intimt förknippad med mättidens längd. Sker färre noteringar om belastningen per tidsenhet erhålles naturligtvis mindre mängd data vid en given total mättid. Dessutom måste försiktighet iaktagas med val av mätfrekvens ur en annan synpunkt. När mätning av exekveringsfrekvensen enligt de beskrivna metoderna i avsnitt 4.1 användes blir varje mätvärde ett m e d e l v ä r d e under den tid som passerat sedan förra avläsningen. Därför kan felaktiga resultat erhållas vid flera av de beskrivna formerna av presentationssätten. Se vidare figurerna 5.1 - 5.3.

I figur 5.1 visas vad som kan hända vid "kvasikontinuerlig" mätning enligt avsnitt 3.0.1 B då avståndet mellan mätvärdesregistreringarna är olika. Den i a) visade kurvan har ett ganska lungt beteende utan vare sig resultat med noll eller 100 procents belastning. b) - kurvan däremot visar ett helt annat beteende trots samma belastning.

Skillnaden är den skilda tidsupplösningen.

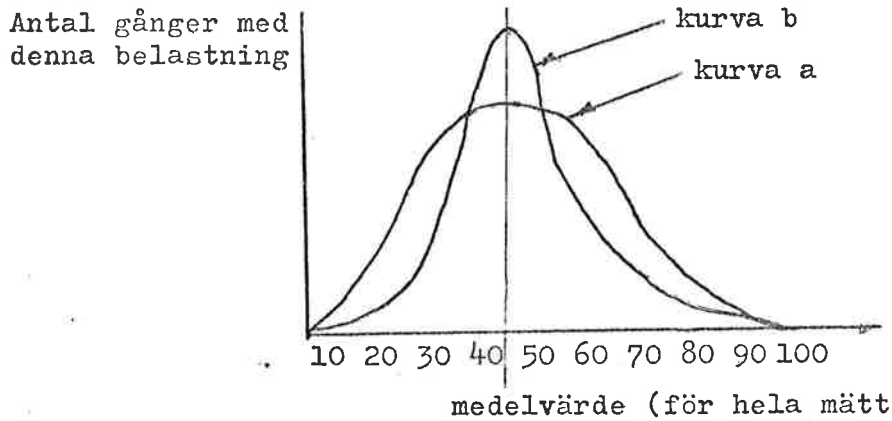


Figur 5.1 a



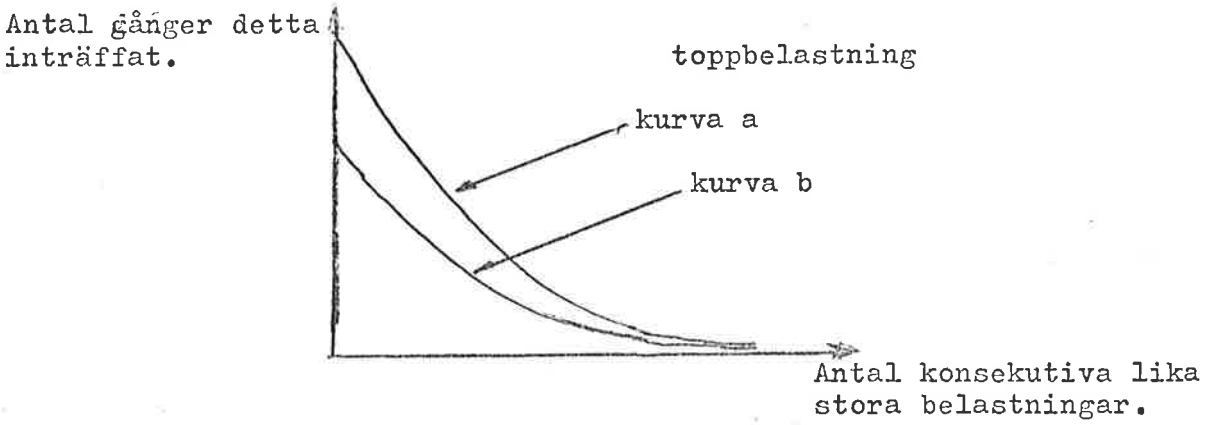
Figur 5.1 b

Även vid registrering av belastningens fördelning blir resultatet beroende av mätfrekvensen, så att tätare mätningar ger större spridning med mindre maximalvärde. Se figur 5.2, kurva a. Större avstånd mellan mätvärdesregistreringarna ger en kurva som alltmer grupperar sig kring medelvärdet fär hela mättiden. Se kurva b.



Figur 5.2

När antalet konsekutiva belastningar av samma storlek uppritas vid olika mätfrkvens syns även där ett klart samband. Tydligast framgår detta vid belastningsnivåer nära max eller min, d.v.s. där mätning kan vara intressant. Ett exempel finns i figur 5.3.



Figur 5.3 Olika kurvor visande två exempel med antal konsekutiva toppbelastningar regisrterade med olika mätfrekvens. Kurva a svarar mot den högsta mätfrkvensen.

Som framgått av figurerna ovan kommer även max och minvärdena att förändras vid olika tidsavstånd mellan mätpunkterna. Tätare mätningar ger betydligt större sannolikhet att de båda extremvärdena noll och 100 procent skall uppnås.

Det optimala värdet på mätfrekvensen blir beroende på flera saker. Övre gränsfrekvensen bestämmes i första hand av hur viktigt det är att kunna följa belastningens alla snabba variationer. Undre gränsfrekvensen utgöres av total mättid. Vid mätning med hög frekvens är det dessutom risk att mätrutiner ger en alltför stor belastning på centralenheten.

Vid val av lämpliga mätintervall får kravet på sann belastning vägas motkravet på att mätrutinen inte skall belasta CPU. Längre mätintervallger dessutom lugnare kurvor som är betydligt enklare att läsa.

Slutligen valet av lämpliga noggrannhetsangivelser, viket endast förekommer vid statistiska mätmetoder. De värden som bör användas beror helt på det önskemål om noggrannhet som uppställts. Större noggrannhet kräver alltid längre mättid. Se vidare avsnitt 3.0.2.

5.5 Starta mätningen

Vid start av mätningen skall mätprogrammet aktiveras. eller yttre enhet startas, beroende på mätmetod. När mätprogram användes måste eventuella parametrar inläsas. Detta sker lämpligen genom att operatören via konsolskrivare inför önskade parametrar.

Baseras mätmetoden på någon inre tidbas i datorn måste denna eventuellt kalibreras. Kalibreringen behöver emellertid normalt endast göras en gång i varje system, såvida inga större ändringar göres som kan tänkas påverka den använda tidbasen.

5.6 Kontrollera att mätningen förlöper normalt.

Det som bör kontrolleras under mätningens gång är främst att den belastningssituation som undersökes verkligen är den önskade. Eventuella oväntade larm och avbrott kan medföra en snedvridning av utresultatet vilket kan medverka till en felbedömning av belastningen.

5.7 Avsluta mätningen

Sker inte mätningen automatiskt genom mätprogrammets försorg avslutas den med ett operatörsingripande. I så fall måste operatören själv hålla reda på total mättid.

Om yttre enheter användes kan mätningen avslutas genom att dessa yttre enheter frånkopplas.

5.8 Ladda och aktivera program som utför slutberäkning och presentation av resultatet.

Det program som tar hand om mätresultatet efter avslutad mätning och omvandlar rsultatet till ett människovänligt presentationssätt skall inkallas.

Mätprogram, där sådant användes, bör efter avslutad mätning läggas tillbaka på massminne för att inte utgöra en onödig belastning på primärminnet.

5.9 Analysera resultatutskrift.

Vid analys av resultatet bör eventuella systematiska fel noggrannt inringas. Vidare måste det som sägs om mätintervallets inverkan på resultatet i avsnitt 5.4 observeras. Genom val av olika mätinervall är det nämligen möjligt att inom vida gränser få fram helt olika synviklar på belastningen.

K A P I T E L 6

VALT LÖSNINGSFÖRSLAG

6 Valt lösningsförslag

Efter diskussion med personal på avdelning YLA, ASEA framkom det att en utveckling av lösningsförslag 4.1.1.2 i första hand var intressant. Förslaget innebär en mätning av totalt utnyttjad CPU-tid utan åtskillnad av operativsystem och applikationsprogram. Mätningen sker under en TOTAL MÄTTID av storleksordningen minuter med avläsningar av belastningens medelvärde under kortare intervall, MÄTINTERVALL.

Det utvecklade mätpaketet består helt av mjukvara i form av FORTRAN - program och är i det aktuella fallet anpassat till MODCOMP, MAX III. Eftersom det är skrivet i FORTRAN kan en anpassning till likvärdiga system av annat fabrikat ganska enkelt göras. Hur detta skall ske beskrives senare.

Kapitel 6 har följande indelning:

6.1 Funktionsbeskrivning av programpaketet.
6.2 Beskrivning av varje program.
6.3 Praktiskt handhavande vid mätning.
6.4 Några mätningar på simulerad belastning.
6.5 Infogning i befintligt system.
6.6 Anpassning till annat system.

De nödvändiga programmen, vilka är fem stycken, är stansade på hålkort och inläses samt katalogiseras lämpligen på disc-fil i det system där de skall användas.

6.1 Funktionsbeskrivning av programpaketet.

Programpaketet består av fem olika realtidsprogram lagrade som separata TASKS. Anledningen till denna uppdelning är det varierande prioritetsbehov som olika delfunktioner av mätprocessen har. De fem programmen är:

1. IDLELO: En enkel räknare på lägsta prioritet. Fungerar som idle-loop och är därmed kärnan i hela mätsystemet.

2. CALIBR: Program som tillsammans med IDLELO kalibrerar aktuell dators cykeltid. Programmet aktiverar själv idle-loopen och frågar operatören om tickfrekvensen för att sedan kalibrera och skriva ut beräknad SKALFAKTOR.(se sidan 4.5)

3. READO: Läser mätvärdena som IDLELO producerar och lagrar dessa i en buffert. Motsvarar avläsningsprogrammet i figur 4.9 på sidan 4.11. Styr dessutom nollställningen av IDLELO.

4. CALCUL: Tömmer ovan angivna buffert och utför vissa beräkningar på mätvärdena för att sedan sortera in dessa i två tabeller. SUPERV tömmer sedan dessa tabeller för slutberäkning och presentation. Motsvarar beräningsprogrammet i figur 4.9 på sidan 4.11.

5. SUPERV: Övervakar och styr IDLELO, READO och CALCUL för att operatörens ingripanden skall vara minimala.Förmedlar dessutom kommunikationen med operatören för inläsning av parametrar och utskrift av mätresultatet. Vid mätning är det endast SUPERV som behöver aktiveras, resten sker automatiskt.

Prioritetsnivåerna för programmen är:

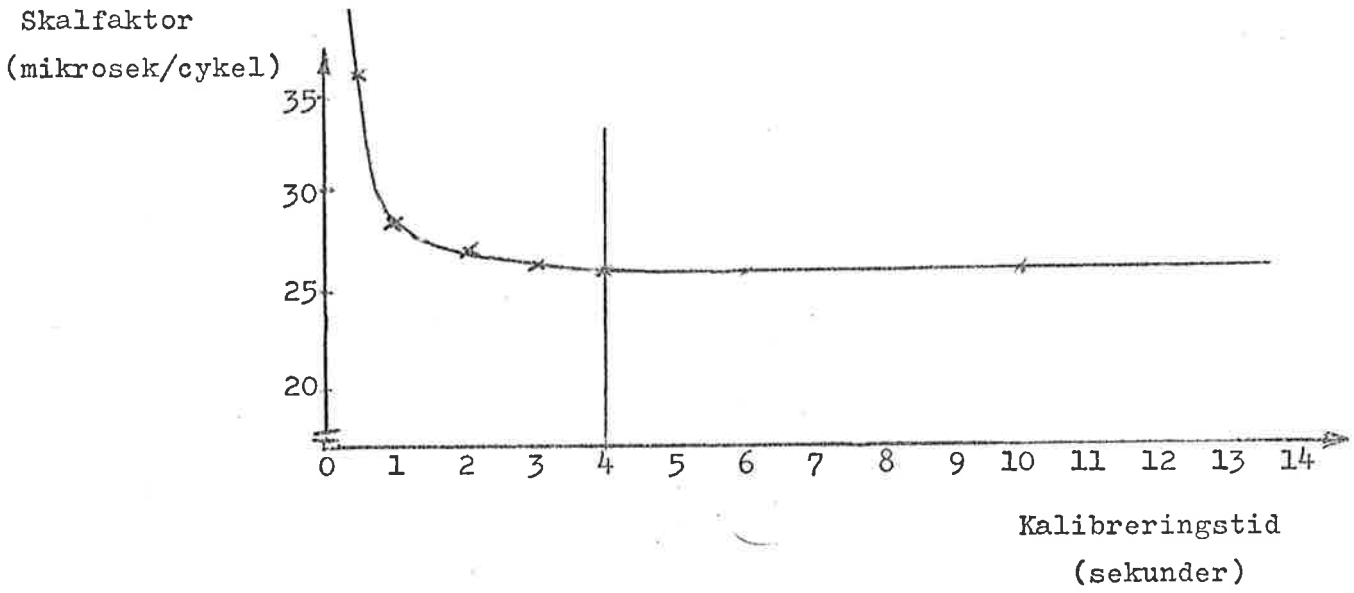
- IDLELO 127
- CALIBR bestäms av operatören, lämpligen 70
- READO 30
- CALCUL 70
- SUPERV bestäms av operatören, lämpligen 90

Prioritetsnivåernas läge är inte på något sätt kritiskt. Viktigt är emellertid att IDLELO har lägst prioritet av samtliga realtidsprogram i systemet. På samma sätt måste READO ha högsta prioritet. Övriga program kan läggas på "mellanprioritet". Önskas ändringar av nivån på de tre programmen som inte styrs av operatören måste vissa ändringar direkt i programinstruktionerna göras. De instruktioner som är aktuella i så fall är de subrutinanrop som aktiverar program, nämligen CALL TSKACT(a,b,c,d). Parameter b anger prioritetsnivån som heltal, övriga parametrar behöver inte ändras.

Vid mätning måste först cykeltiden för loopen kalibreras. Nödvändigheten av kalibrering kan belysas av det faktum att SKALFAKTORN kan variera med en faktor 10 beroende på om datorsystemet är utrustat med hårdvarumässig behandling av flyttal eller inte. Kalibreringen sker genom att aktivera programmet CALIBR som i sin tur gör IDLELO kärnminnesresident och aktivt. Via konsolskrivmaskinen frågar programmet om lagrat värde på tickfrekvensen för systemet är korrekt (TICKRATE). Om inte läses ett nytt värde in.

Kalibreringen startar och pågår i 4 sekunder, därefter skrivs beräknad SKALFAKTOR ut på konsolskrivmaskinen. Beräkningen sker enligt ekvation 4.1 på sidan 4.5 med AD och KR satta till noll. Att kalibrering pågår just i 4 sekunder motiveras av figur 6.1 där skalfaktorn plottats som funktion

av kalibreringstiden. Det framgår att vid 4 sekunder har kurvan planat ut och ännu längre mättid ger inte något bättre värde.



Figur 6.1 Skalfaktorns beroende av kalibreringstiden.

Anledningen till uppträdandet vid korta kalibreringstider är tiden för uppstart och avslut för loopen blir så stor att den inte kan försummas i förhållande till den totala kalibreringstiden.

Under den tid då kalibrering sker får inga andra program vara aktiva eftersom detta skulle ge helt felaktiga resultat. Kalibrering kan alltså inte ske under normal drift. Då skalfaktorn emellertid kan anses vara konstant i ett system, såvida inga större mjuk eller hårdvarumässiga ändringar göres, behöver kalibrering normalt endast göras en gång.

När skalfaktorn för systemet är känd kan en mätning utföras. SUPERV aktiveras då och ställer vissa frågor om uppdatering av några variabla parametrar. De parametrar som kan ändras av operatören är:

- TOTAL MÄTTID (total measuringtime) i minuter.
- MÄTINTERVALL (measuringinterval) i sekunder.
- SKALFAKTOR (scalingfactor) i mikrosekunder per cykel. Denna parameter är den som erhållits vid kalibreringen.
- TICKFREKVENSEN (tickrate) i ticks per sekund. Bestäms vid systemgenereringen.

När parametrarna erhållit de önskade värdena fortsätter SUPERV att styra mätförloppet. Programmen IDLELO, READO och CALCUL göres kärnminnesresidenta. READO läser och beordrar nollställning av IDLELO som nollställer sig själv. Värdet som READO läser lagras i en buffert. Sedan sker nya avläsningar med en frekvens som bestäms av MÄTINTERVALL och när bufferten är full startar READO upp CALCUL. Detta program tömmer bufferten och gör vissa beräkningar enligt ekvation 4.3 på sidan 4.7 plus kontroll av konsekutiv maxbelastning enligt vad som skisserats i avsnitt 3.0.1 E. Mätvärdena sorteras därefter in i två tabeller för belastningens fördelning respektive konsekutiv maxbelastning.

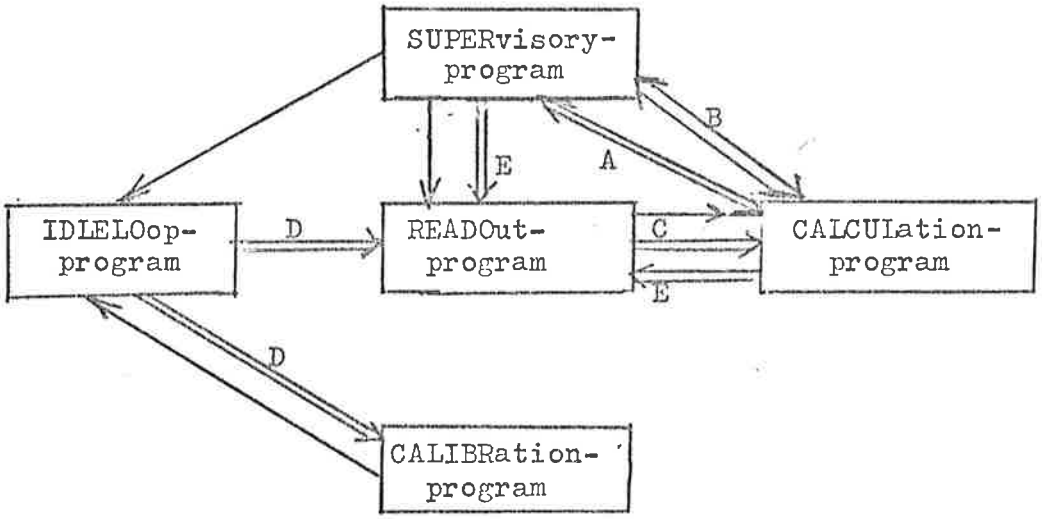
Efter den tid som ges av TOTAL MÄTTID stoppar SUPERV aktiveringen av IDLELO och READO. CALCUL tömmer bufferten för sista gången och SUPERV utför slutberäkning på de värden som finns i de båda tabellerna. Innan dess har dock READO, IDLELO och CALCUL åter gjorts massminnesresidenta. Utskrift av slutresultatet sker på radskrivare.

Kommunikationen mellan programmen sker via en global COMMON - area, som i den befintliga versionen kräver ett utrymme på 440 ord. Eftersom bufferten för lagring av mätvärdena tar största utrymmet, nämligen 400 ord, kan minnesbehovet enkelt minskas genom att skära ner bufferstorleken. Följden blir dock att programmet CALCUL aktiveras oftare

med ökad belastning som följd.

De ändringar som behöver göras är att fältdeklarationen av bufferten som kallas BUFF ändras, samt de tester i programmen READO och CALCUL som testar mot buffertens storlek, alltså 200.

Styrningen av och kommunikationen mellan de olika programmen framgår av figur 6.2.



⟶ Uppstart och avslut.

⟹ Mätvärdes och parameteröverföring.

A) COMMONAREA AA
B) COMMONAREA BB
C) COMMONAREA CC
D) COMMONAREA DD
E) COMMONAREA EE

Figur 6.2 Styrningen av och kommunikationen mellan de olika programmen.

6.2 Beskrivning av varje program.

I detta avsnitt följer ett flödesschema och en listning för varje program. Minnesbehovet för programmen är:

- IDLELO 0.8 kord
- CALIBR 2.4 kord
- READO 0.8 kord
- CALCUL 0.8 kord
- SUPERV 3.4 kord

Vid själva mätningen är dock endast IDLELO, READO och CALCUL kärnminnessresidenta. Alltså blir det residenta minnesbehovet 2.4 kord. Bedöms detta vara för mycket kan CALCUL tillåtas vara massminnesresident under hela mätningen utan några större tidsförluster. I så fall slopas följande två instruktioner i programmet SUPERV:

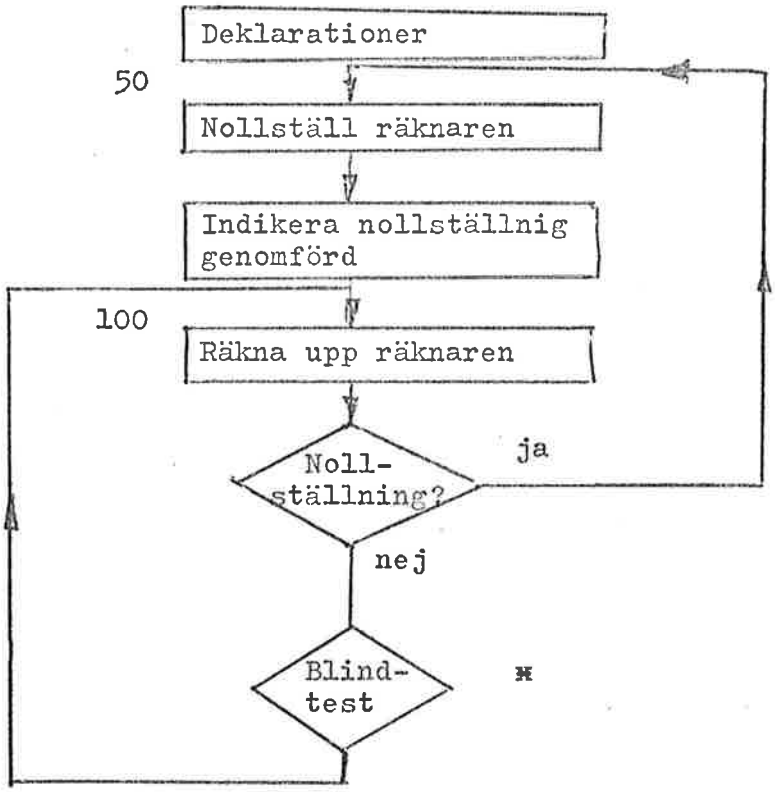
```
CALL ESTABL(IACAN(CAL),90,IACAN(IFIL),L1)
```
(sidan 2 i programlistningen)

```
CALL DEESTA(IACAN(CAL),L7)
```
(sidan 3 i programlistningen)

Det minimala nödvändiga primärminnesutrymmet är alltså 1.6 kord.

IDLE - LOOPPROGRAM

IDLELO

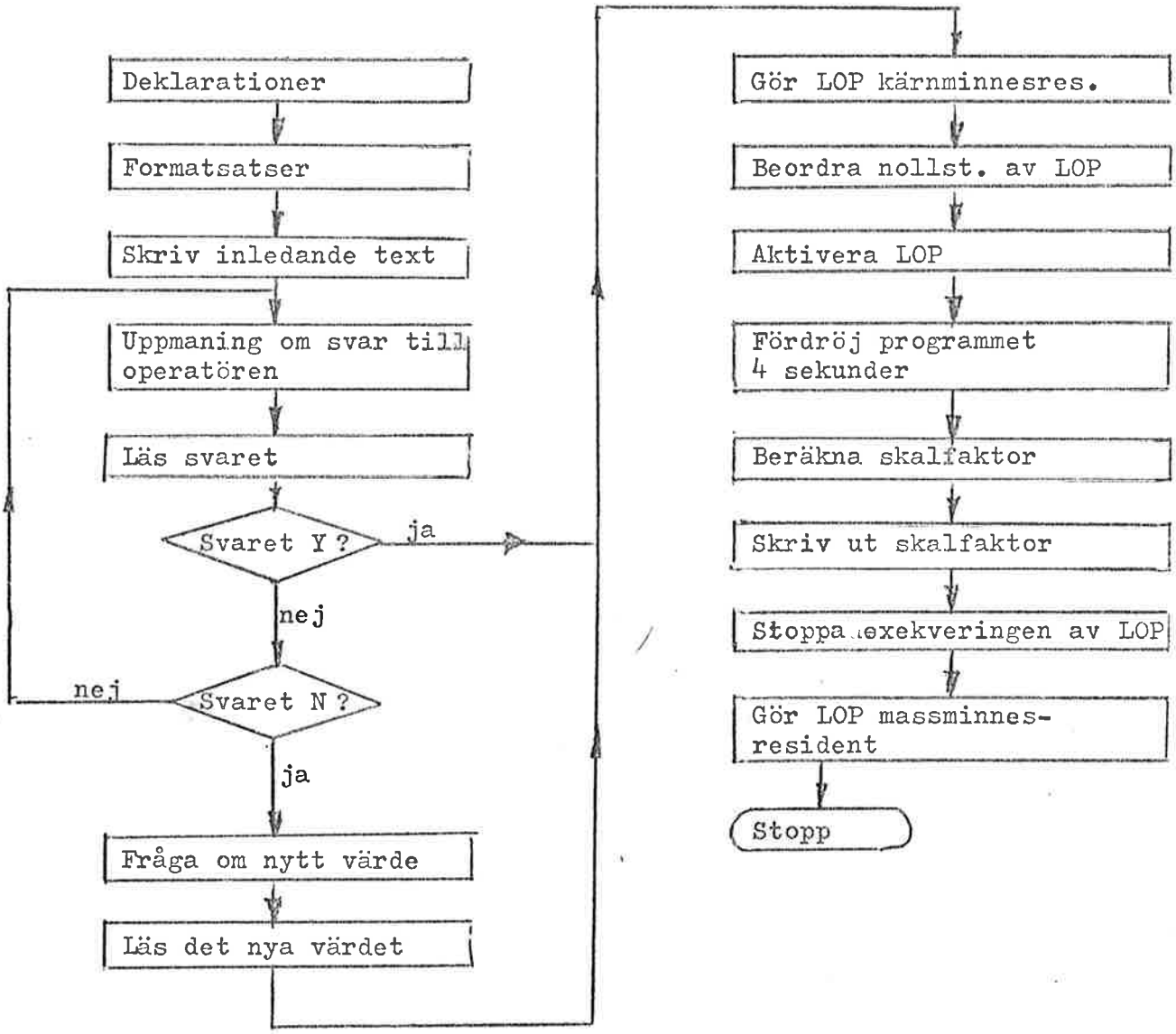


*) Blindtestens enda uppgift är att lura kompilatorn då programmet saknar slut.

```
      PROGRAM IDLELO
C  IDLE-LOOPPROGRAM LOP.
C  THIS PROGRAM IS COUNTING CONTINOUSLY WHILE ITS BEEING EXECUTED.
      INTEGER A,B
C  A AND B ARE USED TO FOOL THE COMPILATOR.
      REAL ILC
C  ILC IS THE COUNTER.
      LOGICAL ZERO
      COMMON  /DD/ILC,ZERO
C  COMMONBLOCK DD MAINTAIN CONTACT WITH THE READOUTPROGRAM REA.
      DATA    A,B/0,1/
   50 CONTINUE
      ILC=0.0
      ZERO=.FALSE.
  100 CONTINUE
      ILC=ILC + 1.0
      IF(.NOT.ZERO) GOTO 100
      IF(A.NE.B) GOTO 50
      STOP
      END
```

KALIBRERINGSPROGRAM

CALIBR

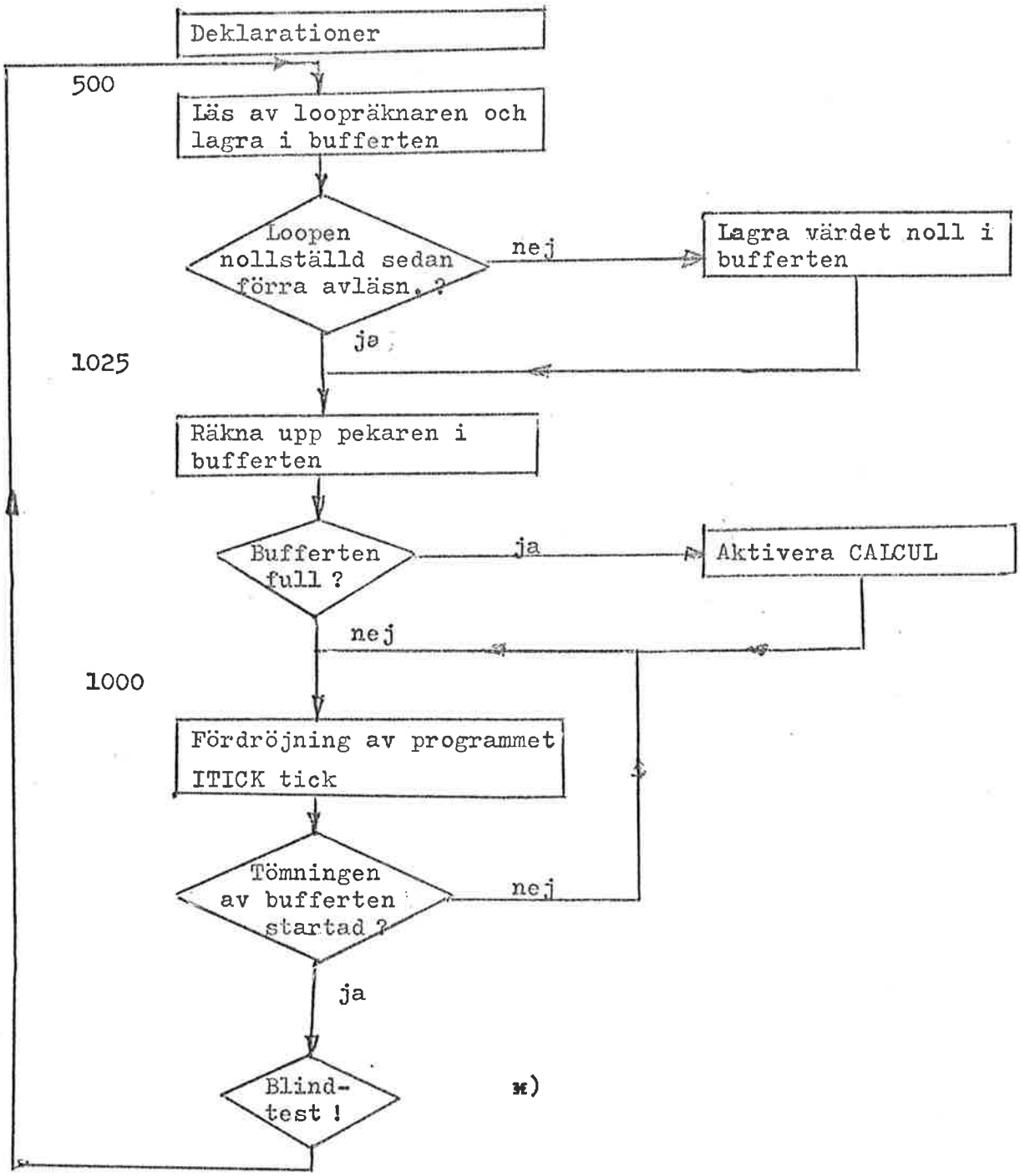


```
      PROGRAM CALIBR
C CALIBRATIONPROGRAM CLB
C  THIS PROGRAM CALIBRATES THE TIME OF THE IDLE-LOOPCYCLE IN PROGRAM LOP.
      INTEGER L1,L2,L3,L4,L5,L6,IFIL(3),LOP(3),IRATE,ANSW
      INTEGER Y,N,CLB(3)
      INTEGER CTIM
C  L1-L6 ARE TESTVARIABLES.
C  L1-L8 ARE TESTVARIABLES.
C  IFIL IS USED FOR STORING THE NAME OF THE INPUTFILE IN ASCII-CODE.
C  LOP IS USED FOR STORING THE NAME LOP IN ASCII-CODE.
C  IRATE IS THE TICKRATE FOR THE SYSTEM.
C  ANSW CONTAINS THE OPERATORS ANSWERS.
C  IDATE CONTAINS DATE AND TIME OF DAY.
C  Y AND N ARE THE ACCEPTABLE ANSWERS FROM THE OPERATOR.
C  CLB IS THE PROGRAMNAME CLB IN ASCII-CODE.
C  CTIM IS THE CALIBRATIONTIME.
      REAL SK,ILC
C  SK IS THE CALCULATED VALUE OF THE SCALINGFACTOR.
C  ILC IS THE VALUE OF THE IDLE-LOOPCOUNTER.
      LOGICAL ZERO
      COMMON /DD/ILC,ZERO
C  COMMONBLOCK DD MAINTAIN THE CONTACT BETWEEN CLB AND LOP.
      DATA    LOP/'L ','O ','P '/
      DATA    IFIL/'L ','M ','T '/
      DATA    Y,N/'Y ','N '/
      DATA    CLB/'C ','L ','B '/
 8000 FORMAT(1HL,5X,15HSCALINGFACTOR= ,F9.3,19H MICROSECONDS/CYCLE///)
 8010 FORMAT(///1X,23HTHE TICKRATE IS SET TO ,I3,31H TICKS/SECOND. IS TH
     *AT CORRECT?)
 8020 FORMAT(1A1)
 8030 FORMAT(1X,11HNEW VALUE: )
 8040 FORMAT(1I3)
 8050 FORMAT(1X,14HANSWER Y OR N:)
C  UPDATING OF IRATE.
      WRITE(1,8010) IRATE
  100 CONTINUE
      WRITE(1,8050)
      READ(1,8020) ANSW
      IF(ANSW.EQ.Y) GOTO 500
      IF(ANSW.EQ.N) GOTO 200
      GOTO 100
  200 CONTINUE
      WRITE(1,8030)
```

```
      READ(1,8040) IRATE
  500 CONTINUE
C  MAKING LOP CORE-RESIDENT AND ACTIVE.
      CALL ESTABL(IACAN(LOP),127,IACAN(IFIL),L1)
      ZERO=.TRUE.
      CALL TSKACT(IACAN(LOP),127,LMT,L2)
      CTIM=IRATE*4
      CALL TSKDLL(0,CTIM,,.FALSE.,.FALSE.)
C  CALCULATION OF THE SCALINGFACTOR.
 1000 CONTINUE
      SK=4000000.0/(ILC*1.035)
      WRITE( 1,8000) SK
C  KILLING AND DEESTABLISH OF LOP.
      CALL TSKKIL(IACAN(LOP),0,L4)
      CALL DEESTA(IACAN(LOP),L6)
      STOP
      END
```

AVLÄSNINGSPROGRAM

READO

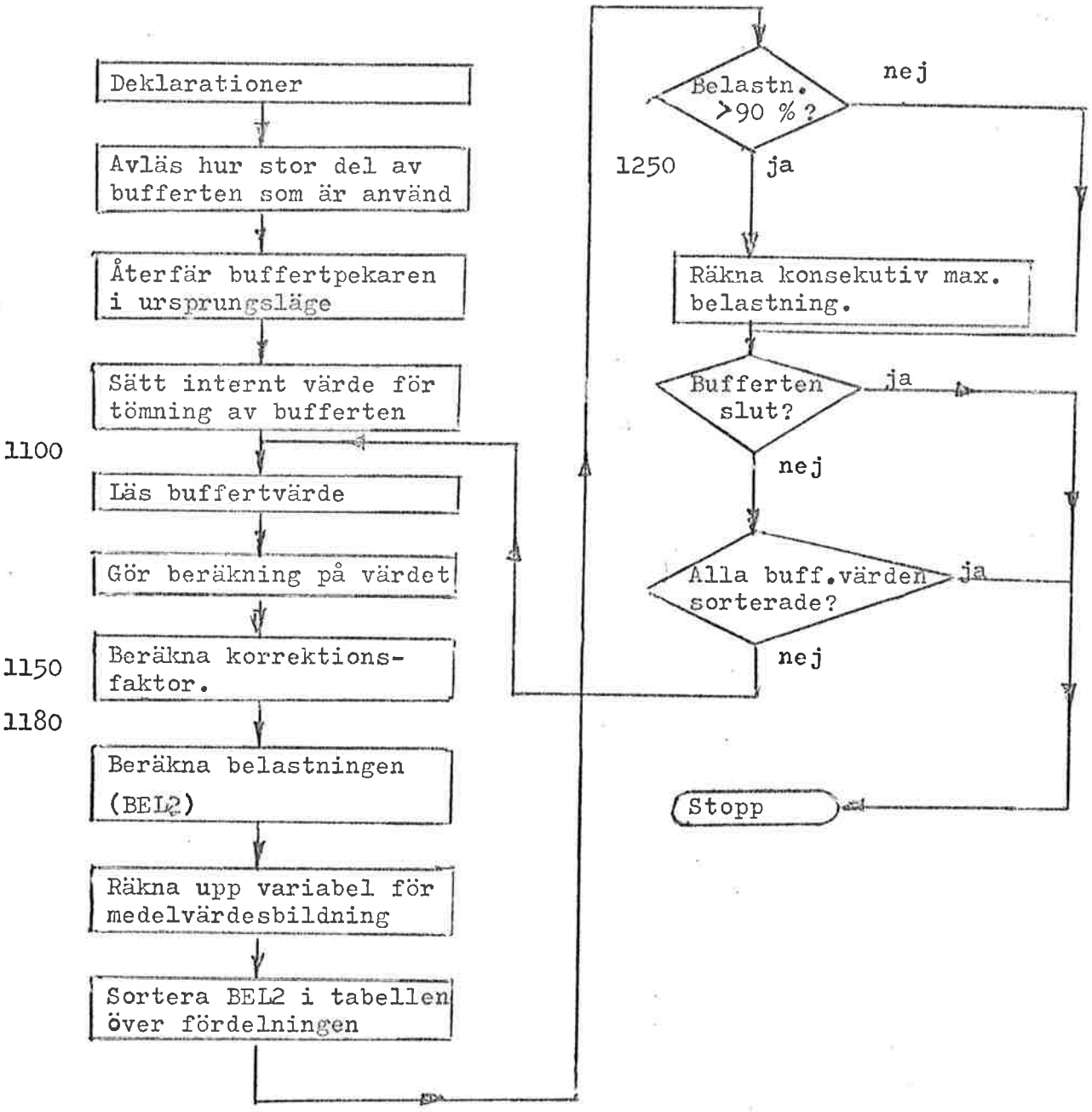


*) Blindtesten användes endast för att lura kompilatorn då programmet saknar slut.

```
      PROGRAM READO
C  READOUTPROGRAM REA
C  THIS PROGRAM IS READING THE VALUES FROM THE IDLE-LOOP ILO AND STORES
C  THEM INTO THE BUFFER BUFF.
      INTEGER Ix,L1,CAL(3),A,B,ITICK
C  IX IS USED FOR INDEXING.
C  L1 IS A TESTVARIABEL IN TSKACT.
C  CAL CONTAINS CAL IN ASCII-CODE.
C  A AND B ARE USED TO FOOL THE COMPILATOR.
C  ITICK IS THE MEASURINGINTERVAL.
      REAL    ILC,BUFF(200)
C  ILC IS THE NAME OF THE IDLE-LOOPCOUNTER.
C  BUFF IS THE BUFFER
      LOGICAL ZERO
      COMMON  /CC/BUFF
      COMMON  /DD/ILC,ZERO
      COMMON  /EE/Ix,ITICK
C  COMMONBLOCK CC MAINTAIN CONTACT WITH CAL, DD WITH LOP AND EE
C  WITH CAL AND SUP.
      DATA    CAL/'C ','A ','L '/
      DATA    A,B/1,0/
  500 CONTINUE
      BUFF(IX)=ILC
      IF(.NOT.ZERO) GOTO 1025
      BUFF(IX)=0.0
 1025 CONTINUE
      IX=IX + 1
      ZERO=.TRUE.
      IF(IX.LE.200) GOTO 1000
      CALL TSKACT(IACAN(CAL),70,LMT,L1)
 1000 CONTINUE
      CALL TSKDEL(0,ITICK,.FALSE.,.FALSE.)
      IF(IX.GT.200) GOTO 1000
      IF(A.NE.B) GOTO 500
      STOP
      END
```

BERÄKNINGSPROGRAM

CALCUL

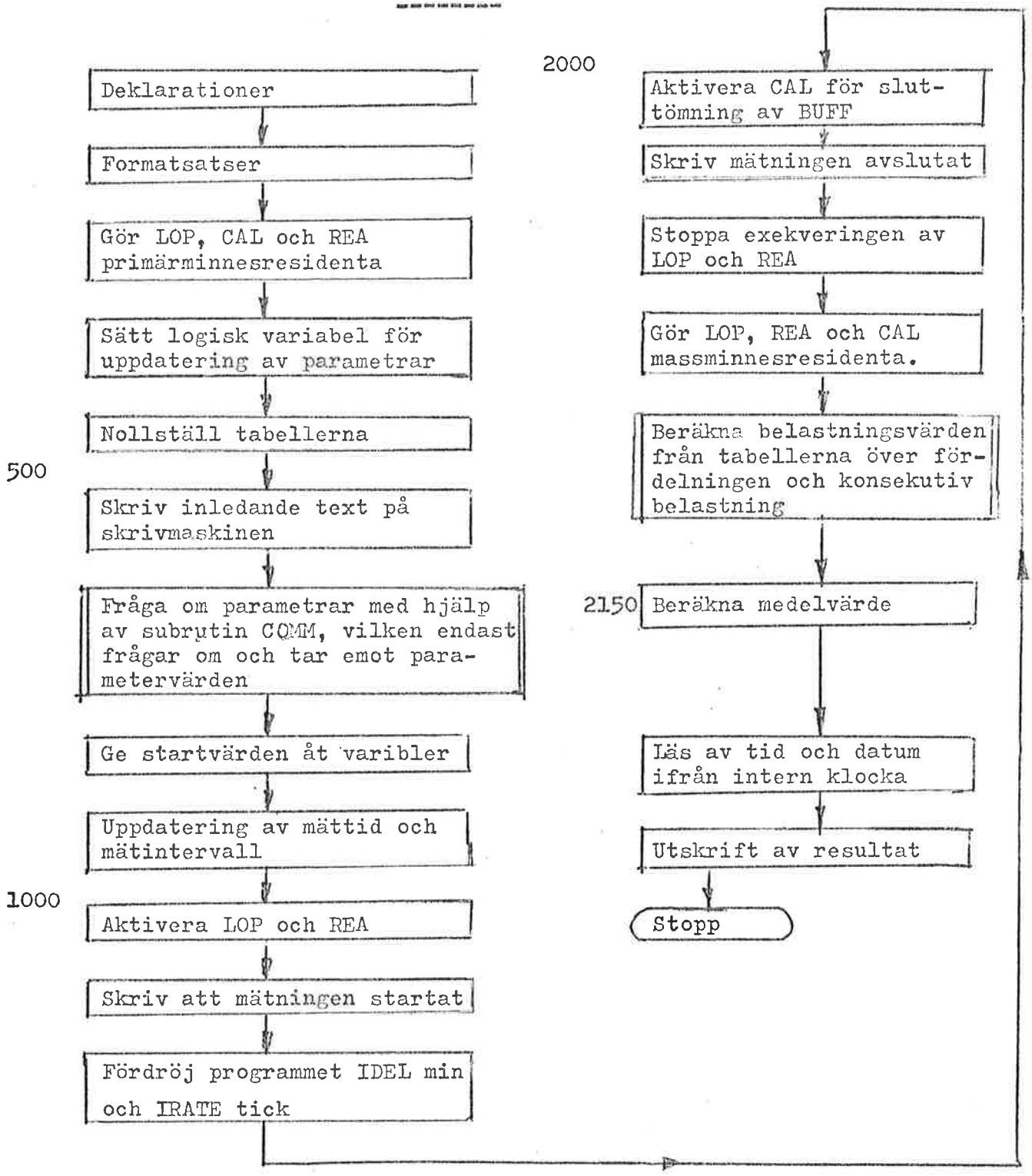


```
      PROGRAM CALCUL
C  CALCULATINGPROGRAM CAL.
C  THIS PROGRAM IS USED FOR CALCULATION OF THE MEASURINGRESULTS.
      INTEGER I,II,ITAB(10),IKON(10),L,IX,ITICK
      INTEGER J
C  I AND II ARE USED FOR INDEXING.
C  IBUFF CONTAINS THE MEASURINGVALUES FROM REA.
C  ITAB AND IKON ARE THE SORTED AND CALCULATED VALUES GIVEN TO SUP.
C  L IS USED FOR COUNTING CONSECUTIVE LOADS.
C  IX IS USED IN REA FOR INDEXING.
C  ITICK IS USED IN REA.
C  J IS THE MAXIMAL INDEX USED IN BUFF.
      REAL    PARA(5),MED,CLOCK,BEL1,BEL2,KORR,B2,MVAL
      REAL    BUFF(200)
C  PARA ARE THE PARAMETERS GIVEN FROM THE OPERATOR VIA SUP.
C  MED ARE USED FOR CALCULATING THE MEANVALUE OF THE CPU-LOAD.
C  CLOCK CONTAINS CLOCKPULSES/SECOND.
C  BEL1 AND BEL2 ARE THE LOAD WITHOUT AND WITH CORRECTION.
C  KORR IS A CORRECTIONFACTOR.
C  B2 IS USED FOR SORTING BEL2 INTO THE TABLE ITAB.
C  MVAL ARE THE VALUES TAKEN FROM IBUFF.
C  BUFF CONTAINS THE MEASURINGVALUES FROM REA.
      COMMON  /AA/PARA
      COMMON  /BB/ITAB,IKON,MED,CLOCK,L
      COMMON  /CC/BUFF/EE/IX,ITICK
C  COMMONBLCK AA AND BB ARE USED FOR COMMUNICATION BETWEEN CAL AND SUP
C  AND CC AND EE BETWEEN CAL AND REA.
      J=IX - 1
      IX=2
      I=2
C
C  CALCULATION PART.
C
 1100 CONTINUE
      MVAL=BUFF(I)
      BEL1=MVAL*PARA(4)/(PARA(2)*1000000.0)
      IF(BEL1.LE.0.5) GOTO 1150
      KORR=1.5 - BEL1*1.5
      GOTO 1180
 1150 CONTINUE
      KORR=1.0 - BEL1*0.5
 1180 CONTINUE
      BEL2=(1.0-(BEL1+PARA(5)*KORR*(PARA(3)/1000000.0)))*100.0
```

```
      IF(BEL2.GT.0.0) MED=MED + BEL2
C  SORTING BEL2 INTO THE RIGHT CLASS.
      B2=BEL2/10.0001
      II=INT(B2) + 1
      ITAB(II)=ITAB(II) + 1
C  CHECKING IF THE LOAD IS MAXIMUM AND IF SO, COUNTING THE CONSECUTIVE
C  MAXIMUM LOADS INTO TABLE IKON.
      IF(II.NE.10) GOTO 1200
      L=L + 1
      GOTO 1250
 1200 CONTINUE
      IF(L.EQ.0) GOTO 1250
      IF(L.LE.10) GOTO 1220
      IKON(10)=IKON(10) + L
      GOTO 1230
 1220 CONTINUE
      IKON(L)=IKON(L) +L
 1230 CONTINUE
      L=0
 1250 CONTINUE
      I=I + 1
      IF(I.GT.200) STOP
      IF(I.LE.J) GOTO 1100
      STOP
      END
```

ÖVERVAKANDE PROGRAM

SUPERV



```
      PROGRAM SUPERV
C  SUPERVISORY PROGRAM SUP
C  THIS PROGRAM IS CONTROLLING THE THREE PROGRAMS LOP, REA AND CAL.
C  ITS ALSO TAKING CARE OF ALL THE NESSESARY I/O-COMMUNICATIONS
      INTEGER ITAB(10),IKON(10),SUP(3),LOP(3),CAL(3),REA(3),IFIL(3)
      INTEGER IDATE(6),HOUR,I,J,L,IX,L1,L2,L3,L4,L5,L6,L7,L8
      INTEGER IDEL,IRATE,ITICK
C  ITAB AND IKON ARE USED FOR STORING MEASURINGVALUES
C  SUP, LOP, CAL AND REA ARE USED FOR STORING PROGRAMNAMES IN ASCII-CODE
C  IFIL CONTAINS THE NAME OF THE INPUTFILE IN ASCII-CODE.
C  IDATE CONTAINS TIME OF DAY AND DATE.
C  HOUR IS THE TIME OF DAY.
C  I AND J ARE USED FOR INDEXING.
C  L IS USED IN PROGRAM CAL.
C  IX IS AN INDEX IN LOP.
C  THE INTEGERS L1-L8 ARE TESTVARIABLES FROM SUBROUTINECALLS.
C  IDEL IS PARA(1) AS INTEGER.
C  IRATE IS THE NUBER OF CLOCKTICKS IN TSKDEL.
C  ITICK IS PARA(2) IN TICKS.
      REAL    PARA(5),RTAB(10),RKON(10),SVAR,MED,RSUM,Y,CLOCK
C  PARA HOLDS THE PARAMETERS FROM OPERATOR.
C  RTAB AND RKON CONTAINS THE REULTS TO BE  WRITTEN.
C  SVAR HOLDS THE ANSWERS Y OR N FROM THE OPERATOR.
C  MED IS THE AVERAGEVALUE OF THE CPU-LOAD.
C  RSUM IS THE TOTAL NUMBERS OF MEASURINGVALUES.
C  Y IS USED FOR DIVIDING MINUTES INTO MINUTES AND TICKS.
C  CLOCK TRANSFERS IDATE(6) TO THE PROGRAM CAL VIA COMMONBLOCK BB.
      LOGICAL UPDAT
C  UPDAT INDICATES IF THE MEASURINGINTERVAL HAS BEEN CHANGED.
      COMMON  /AA/PARA
      COMMON  /BB/ITAB,IKON,MED,CLOCK,L
      COMMON  /EE/IX,ITICK
C  COMMONBLOCK AA AND BB ARE USED FOR COMMUNICATION BETWEEN SUP AND CAL.
C  AND EE BETWEEN SUP, REA AND LOP.
      DATA    SUP/'S ','U ','P '/,LOP/'L ','O ','P '/
      DATA    CAL/'C ','A ','L '/,REA/'R ','E ','A '/
      DATA    IFIL/'L ','M ','T '/
 8000 FORMAT(1H1///1X,48HTHIS IS A PROGRAM FOR MEASURING OF THE CPU-LOAD
     *,/1X,54HDO YOU WISH TO CHANGE ANY OF THE FOLLOWING PARAMETERS./1X,
     *35HIN THAT CASE ANSWER Y, OTHERWISE N.//)
 8020 FORMAT(1X,20HTOTAL MEASURINGTIME ,F5.1,10H MINUTES? )
 8030 FORMAT(1X,16HMEASURINGINTERVAL ,F5.2,10H SECONDS? )
 8050 FORMAT(1X,14HSCALINGFACTOR ,F7.2,21H MICROSECONDS/CYCLE? )
```

```
 8060 FORMAT(1X,9HTICKRATE ,F5.1,14H TICKS/SECOND.)
 8070 FORMAT(///1X,26HTHE MEASURING HAS STARTED.)
 8080 FORMAT(///1X,17HEND OF MEASURING.)
 8100 FORMAT(1H1,15X,30HRESULTS OF CPU-LOADMEASURING: ,6H DATE ,I4,1H-,
     *I2,1H-,I2,3X,12HTIME OF DAY ,I2,1H.,I2/16X,66(1H*)//)
 8104 FORMAT(1X, 97HTHE CPU-LOAD IS MEASURED WITH THE USE OF AN IDLE-LOO
     *PCOUNTER AND THE LOAD-FIGURES BELOW SHOWS THE/1X,60HTOTAL LOAD OF
     *THE MONITOR AND THE USER-PROGRAMS ALLTOGETHER.///)
 8108 FORMAT(1X,33HTHE DISTRIBUTION OF THE CPU-LOAD://)
 8110 FORMAT(1X,15HLOAD IN PERCENT,13X,10(2X,I2,1H-,2X))
 8115 FORMAT(1H+,28X,9(5X,I2),5X,I3/)
 8120 FORMAT(1X,23HDISTRIBUTION IN PERCENT,5X,10(2X,F5.1)//)
 8130 FORMAT(1X,13HAVERAGE LOAD:,F6.2,8H PERCENT////)
 8140 FORMAT(1X,52HTHE DISTRIBUTION OF CONSECUTIVE 90-100 PERCENT LOAD:)
 8150 FORMAT(//1X,27HNUMBER OF CONSECUTIVE LOADS   ,9(5X,I2),5X,I2,1H-/)
 8160 FORMAT(//1X,28HTHE TOTAL MEASURINGTIME WAS ,F5.1,14H MINUTES WITH
     *,F5.2,5HH SECONDS BETWEEN THE READINGS OF THE IDLE-LOOPCOUNTER.)
C
C  THE FOLLOWING SUBROUTINE-CALLS MAKE THE PROGRAMS LOP, CAL AND REA
C  CORE-RESIDENT.
      CALL ESTABL(IACAN(LOP),127,IACAN(IFIL),L1)
      CALL ESTABL(IACAN(CAL),90,IACAN(IFIL),L1)
      CALL ESTABL(IACAN(REA),30,IACAN(IFIL),L1)
      UPDAT=.FALSE.
C  SETTING ITAB AND IKON TO ZERO.
      DO 500 I=1,10
      ITAB(I)=0
      IKON(I)=0
  500 CONTINUE
C  THEASE LINES TOGETHER WITH THE SUBROUTINE COMM ESTABLISH CONTACT WITH
C  THE OPERATOR FOR QUESTIONS ABOUT SOME CHANGABLE PARAMETERS.
      WRITE( 1,8000)
      WRITE( 1,8020) PARA(1)
      CALL COMM(1,UPDAT,PARA)
      WRITE(1,8030) PARA(2)
      CALL COMM(2,UPDAT,PARA)
C  PARA(3) IS A CORRECTIONFACTOR.
      PARA(3)=25.0
      WRITE( 1,8050) PARA(4)
      CALL COMM(4,UPDAT,PARA)
      WRITE(1,8060) PARA(5)
      CALL COMM(5,UPDAT,PARA)
C  GIVING STARTVALUES.
```

```
      IX=2
      L=0
      MED=0.0
C  UPDATING OF THE MEASURINGTIME AND INTERVAL.
      IF(.NOT.UPDAT) GOTO 1000
      CLOCK=PARA(5)
      Y=DIM(PARA(1),AINT(PARA(1)))
      IDEL=INT(PARA(1))
      IRATE=INT(Y*60.0*PARA(5))
      ITICK=INT(PARA(2)*PARA(5))
 1000 CONTINUE
C  ACTIVATION OF LOP AND REA. REACTIVATION OF SUP AFTER PARA(1) MINUTES
      CALL TSKACT(IACAN(LOP),127,LMT,L2)
      CALL TSKACT(IACAN(REA),30,LMT,L3)
      WRITE(1,8070)
      CALL TSKDEL(IDEL,IRATE,.FALSE.,.FALSE.)
C
 2000 CONTINUE
C  ACTIVATION OF CAL FOR THE LAST EMPTYING OF THE BUFFER BUFF.
      CALL TSKACT(IACAN(CAL),70,LMT,L3)
C  KILLING OF LOP AND REA. DEESTABLISHING OF LOP, CAL AND REA.
      WRITE(1,8080)
      CALL TSKKIL(IACAN(LOP),0,L4)
      CALL TSKKIL(IACAN(REA),0,L5)
      CALL DEESTA(IACAN(LOP),L6)
      CALL DEESTA(IACAN(REA),L7)
      CALL DEESTA(IACAN(CAL),L8)
C
C  THE FINAL CALCULATION BEFORE PRESENTATION.
      IF(L.EQ.0) GOTO 2050
      IF(L.LE.10) GOTO 2020
      IKON(10)=IKON(10) + L
      GOTO 2030
 2020 CONTINUE
      IKON(L)=IKON(L) + L
 2030 CONTINUE
 2050 CONTINUE
      RSUM=0.0
      DO 2100 I=1,10
      RTAB(I)=FLOAT(ITAB(I))
      RSUM=RSUM + RTAB(I)
 2100 CONTINUE
      DO 2150 I=1,10
```

```
      RTAB(I)=(RTAB(I)/RSUM)*100.0
 2150 CONTINUE
      MED=MED/RSUM
      DO 2200 I=1,10
      RKON(I)=FLOAT(IKON(I))
      RKON(I)=(RKON(I)/RSUM)*100.0
 2200 CONTINUE
C  RKON IS NOW HOLDING VALUES OF CONSECUTIVE LOADS.
C  EXTRACTING HOUR ON DAY OUT OF IDATE(1) FROM TIMDAT.
      CALL TIMDAT(IDATE)
      HOUR=IDATE(1)/60
      IDATE(1)=IDIM(IDATE(1),HOUR*60)
C  HOUR AND IDATE(1) ARE NOW CONTAINING HOURS AND MINUTES RESPECTIVILY.
C  PRESENTATION OF THE RESULTS.
      WRITE(3,8100) IDATE(5),IDATE(3),IDATE(4),HOUR,IDATE(1)
      WRITE(3,8104)
      WRITE(3,8108)
      WRITE(3,8110) (I1, I1=0,90,10)
      WRITE(3,8115) (I2, I2=10,100,10)
      WRITE(3,8120) (RTAB(I), I=1,10)
      WRITE(3,8130) MED
      WRITE(3,8140)
      WRITE(3,8150) (I, I=1,10)
      WRITE(3,8120) (RKON(I), I=1,10)
      WRITE(3,8160) PARA(1),PARA(2)
      STOP
      END
```

```
      SUBROUTINE COMM(I,UPDAT,PARA)
C  THIS SUBROUTINE DETERMINES WHEATHER THE OPERATORS ANSWER IS J OR NO AND
C  THEREAFTER MAKES THE NESSESARY OPERATIONS.
      INTEGER I,Y,N,SVAR
C  I IS USED FOR INDEXING.
C  Y AND N ARE THE ACCEPTABLE ANSWERS FROM THE OPERATOR.
C  SVAR IS THE OPERATORS ANSWER.
      REAL    PARA(5)
C  PARA  AND UPDAT ARE USED IN THE MAINPROGRAM.
      LOGICAL UPDAT
      DATA    Y,N/'Y ','N '/
 8500 FORMAT(1A1)
 8510 FORMAT(1X,15HANSWER Y OR N: )
 8520 FORMAT(F7.2)
 8530 FORMAT(1X,11HNEW VALUE: )
 8540 FORMAT(/)
 5000 CONTINUE
C  TESTING THE OPERATORS ANSWER.
      READ( 1,8500) SVAR
      IF(SVAR.EQ.Y) GOTO 5100
      IF(SVAR.EQ.N) RETURN
      WRITE( 1,8510)
      GOTO 5000
 5100 CONTINUE
      WRITE( 1,8530)
      READ( 1,8520) PARA(I)
      WRITE(1,8540)
C  UPDATING OF THE LOGICAL VARIABLE UPDAT.
      IF(I.NE.4) UPDAT=.TRUE.
      RETURN
      END
```

6.3 Praktiskt handhavande vid mätning.

Vid mätning respektive kalibrering behöver endast ett program aktiveras av operatören. Resten sker automatiskt. I båda fallen måste dock operatören svara några frågor om använda parametrar vid mätning och beräkning. Dessa frågor ställs via skrivmaskin .

KALIBRERING

Under den tid kalibreringen pågår får inga andra funktioner som belastar CPU pågå. Detta ger i så fall för små värden på SKALFAKTORN.
Handhavande:

1. Aktivera CALIBR med kommandot

   /CLB/EXE (prioritet) (fil)

   Prioriteten sättes exempelvis till 70.
   Fil anger den discfil där programmet finns lagrat.

2. Enheten TY (skrivmaskin) skriver ut texten enligt bilaga 1a. Svara på frågan som ställs.

3. Om svaret var N på frågan i 2 skall nytt värde på tickfrekvensen anges. Detta sker som heltal med tre siffror. Bilaga 1b.

4. Efter 4 sekunder skrivs skalfaktorn ut enligt bilaga 1c.

5. Slut

(Bilagan 1 a - c finns på sidan 6.27)

MÄTNING

De fyra parametrarna som skall anges kommenteras nedan:

- MEASURINGTIME måste väljas så lång att den tid under vilken mätning pågår verkligen hinner bli representativ för belastningssituationen. Väljes en kort tid, samtidigt som MEASURINGINTERVAL är lång, blir antalet mätvärden begränsade. Detta kan ge osäkerhet i bedömning av resultatet.
- MEASURINGINTERVAL bör inte väljas alltför kort eftersom detta ger en ökad belastning på centralenheten. Väljes tiden för lång försvinner möjligheten till momentan uppföljning av belastningen och resultatet samlas alltmer kring ett medelvärde. Lämpliga värden torde ligga i området 0.1 - 10 sekunder. Se vidare i avsnitt 5.4 på sidorna 5.3 - 5.6.
- SCALINGFACTOR skall vara den vid kalibreringen erhållna.
- TICKRATE skall vara systemets tickfrekvens.

Handhavande:

1. Aktivera SUPERV med kommandot

   /SUP/EXE (prioritet) (fil)

   Prioriteten sättes exempelvis till 90.
   Fil anger den discfil där programmat finns lagrat.

2. Enheten TY (skrivmaskin) skriver ut texten enligt bilaga 2a. Svara på frågorna och ange nya parametervärden om såönskas. Reella tal med decimalpunkt.

3. THE MEASURING HAS STARTED skrivs ut och mätning pågår så länge som anges av MEASURINGTIME.

4. Efter mättidens slut skrivs texten enligt bilaga 2b ut på TY. Samtidigt skrivs slutresultatet enligt bilaga 3 ut på enheten LP (radskrivare).

5. Slut.

(Bilaga 2 a - c finns på sidan 6.28 och bilaga 3 på sidan 6.29.)

```
III/B  /
ØCLB/EXE 70 LMT
```

1a

```
THE TICKRATE IS SET TO 200 TICKS/SECOND. IS THAT CORRECT?
ANSWER Y OR N:
N
NEW VALUE:
100
```

1b

```
     SCALINGFACTOR=    358.012 MICROSECONDS/CYCLE
```

1c

```
!ABORT (     573D) /LOP/LOP BY   CLB
```

BILAGA 2

```
III/B  /
ØSUP/EXE 90 LMT
```

2a

```
THIS IS A PROGRAM FOR MEASURING OF THE CPU-LOAD.
DO YOU WISH TO CHANGE ANY OF THE FOLLOWING PARAMETERS.
IN THAT CASE ANSWER Y, OTHERWISE N.

TOTAL MEASURINGTIME    1.0 MINUTES?
Y
NEW VALUE;
2.0

MEASURINGINTERVAL  1.00 SECONDS?
N
SCALINGFACTOR  357.98 MICROSECONDS/CYCLE?
N
TICKRATE 100.0 TICKS/SECOND.
N

THE MEASURING HAS STARTED.
!ABORT (    3B1A) /REA/REA BY  SUP
```

2b

```
END OF MEASURING.
!ABORT (    5CBD) /LOP/LOP BY  SUP
```

RESULTS OF CPU-LOADMEASURING: DATE 1975- 9-15 TIME OF DAY 12.27
****

THE CPU-LOAD IS MEASURED WITH THE USE OF AN IDLE-LOOPCOUNTER AND THE LOAD-FIGURES BELOW SHOWS THE TOTAL LOAD OF THE MONITOR AND THE USER-PROGRAMS ALLTOGETHER.

THE DISTRIBUTION OF THE CPU-LOAD:

LOAD IN PERCENT	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
DISTRIBUTION IN PERCENT	54.8	0.0	0.0	0.0	3.2	3.2	0.0	0.0	0.0	38.7

AVERAGE LOAD: 41.93 PERCENT

THE DISTRIBUTION OF CONSECUTIVE 90-100 PERCENT LOAD:

NUMBER OF CONSECUTIVE LOADS	1	2	3	4	5	6	7	8	9	10-
DISTRIBUTION IN PERCENT	0.0	0.0	0.0	0.0	0.0	38.7	0.0	0.0	0.0	0.0

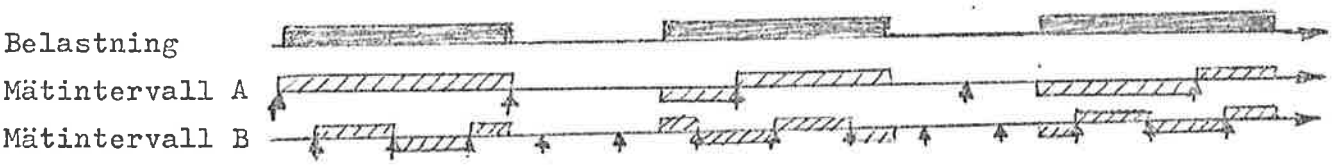
THE TOTAL MEASURINGTIME WAS 0.5 MINUTES WITH 1.00 SECONDS BETWEEN THE READINGS OF THE IDLE-LOOPCOUNTER.

6.4 Några mätningar på simulerade belastningar.

I detta avsnitt kommer ett antal mätningar på olika simulerade belastningar att kommenteras. Först visas hur mätpaketets egen belastning varierar med valet av MEASURINGINTERVAL. Därefter följer några mätningar på en simulerad belastning, där MEASURINGINTERVAL har ändrats för att kunna erhålla olika grad av information om den konsekutiva maxbelastningen.

Vid mätning enligt sidan 6.31 och 6.32 på obelastad dator syns hur medelbelastningen kraftigt ökar vid korta mätintervall. Med intervallet 0.1 sekunder är medelbelastningen 8.52 procent medan ett värde på 1.0 sekunder endast ger 0.04 procent. Vid ännu längre mätintervall minskar inte belastningen nämvärt utan stannar på ett värde av storleksordningen några tiondels promille.

Belastningen som simulerats i mätningarna på sidorna 6.33 - 6.35 belägger CPU under cirka 6 sekunder för att sedan göra en paus på ungefär 4 sekunder innan ny beläggning av CPU sker o.s.v. Att medelbelastningen skiljer sig med några procent beror i första hand på den korta mättiden och hur mätintervallen ligger i förhållande till belastningens variationer. Se figur 6.3



Figur 6.3 Anledningen till att belastningsresultatet kan variera vid korta mättider beroende på mätintervallets läge i förhållande till belastningen.

Notera hur fördelningen för konsekutiv maxbelastning kan styras genom mätintervallets längd för att erhålla bästa möjliga upplösning.

RESULTS OF CPU-LOADMEASURING: DATE 1975- 9-15 TIME OF DAY 12. 5
****

THE CPU-LOAD IS MEASURED WITH THE USE OF AN IDLE-LOOPCOUNTER AND THE LOAD-FIGURES BELOW SHOWS THE TOTAL LOAD OF THE MONITOR AND THE USER-PROGRAMS ALLTOGETHER.

THE DISTRIBUTION OF THE CPU-LOAD:

LOAD IN PERCENT	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
DISTRIBUTION IN PERCENT	99.4	0.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

AVERAGE LOAD: 8.52 PERCENT

THE DISTRIBUTION OF CONSECUTIVE 90-100 PERCENT LOAD:

NUMBER OF CONSECUTIVE LOADS	1	2	3	4	5	6	7	8	9	10-
DISTRIBUTION IN PERCENT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

THE TOTAL MEASURINGTIME WAS 1.0 MINUTES WITH 0.10 SECONDS BETWEEN THE READINGS OF THE IDLE-LOOPCOUNTER.

RESULTS OF CPU-LOADMEASURING: DATE 1975- 9-15 TIME OF DAY 12. 7
****

THE CPU-LOAD IS MEASURED WITH THE USE OF AN IDLE-LOOPCOUNTER AND THE LOAD-FIGURES BELOW SHOWS THE TOTAL LOAD OF THE MONITOR AND THE USER-PROGRAMS ALLTOGETHER.

THE DISTRIBUTION OF THE CPU-LOAD:

LOAD IN PERCENT	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
DISTRIBUTION IN PERCENT	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

AVERAGE LOAD: 0.04 PERCENT

THE DISTRIBUTION OF CONSECUTIVE 90-100 PERCENT LOAD:

NUMBER OF CONSECUTIVE LOADS	1	2	3	4	5	6	7	8	9	10-
DISTRIBUTION IN PERCENT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

THE TOTAL MEASURINGTIME WAS 1.0 MINUTES WITH 1.00 SECONDS BETWEEN THE READINGS OF THE IDLE-LOOPCOUNTER.

RESULTS OF CPU-LOADMEASURING: DATE 1975- 9-15 TIME OF DAY 12.30

****

THE CPU-LOAD IS MEASURED WITH THE USE OF AN IDLE-LOOPCOUNTER AND THE LOAD-FIGURES BELOW SHOWS THE TOTAL LOAD OF THE MONITOR AND THE USER-PROGRAMS ALLTOGETHER.

THE DISTRIBUTION OF THE CPU-LOAD:

LOAD IN PERCENT	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
DISTRIBUTION IN PERCENT	59.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	38.0

AVERAGE LOAD: 41.36 PERCENT

THE DISTRIBUTION OF CONSECUTIVE 90-100 PERCENT LOAD:

NUMBER OF CONSECUTIVE LOADS	1	2	3	4	5	6	7	8	9	10-
DISTRIBUTION IN PERCENT	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	38.0

THE TOTAL MEASURINGTIME WAS 1.0 MINUTES WITH 0.30 SECONDS BETWEEN THE READINGS OF THE IDLE-LOOPCOUNTER.

RESULTS OF CPU-LOADMEASURING: DATE 1975- 9-15 TIME OF DAY 12.25
****

THE CPU-LOAD IS MEASURED WITH THE USE OF AN IDLE-LOOPCOUNTER AND THE LOAD-FIGURES BELOW SHOWS THE TOTAL LOAD OF THE MONITOR AND THE USER-PROGRAMS ALLTOGETHER.

THE DISTRIBUTION OF THE CPU-LOAD:

LOAD IN PERCENT	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
DISTRIBUTION IN PERCENT	59.5	0.0	0.0	0.0	3.3	2.5	0.0	0.0	0.0	34.7

AVERAGE LOAD: 37.54 PERCENT

THE DISTRIBUTION OF CONSECUTIVE 90-100 PERCENT LOAD:

NUMBER OF CONSECUTIVE LOADS	1	2	3	4	5	6	7	8	9	10-
DISTRIBUTION IN PERCENT	0.0	0.0	0.0	0.0	0.0	34.7	0.0	0.0	0.0	0.0

THE TOTAL MEASURINGTIME WAS 2.0 MINUTES WITH 1.00 SECONDS BETWEEN THE READINGS OF THE IDLE-LOOPCOUNTER.

RESULTS OF CPU-LOADMEASURING: DATE 1975- 9-15 TIME OF DAY 12.28
****

THE CPU-LOAD IS MEASURED WITH THE USE OF AN IDLE-LOOPCOUNTER AND THE LOAD-FIGURES BELOW SHOWS THE TOTAL LOAD OF THE MONITOR AND THE USER-PROGRAMS ALLTOGETHER.

THE DISTRIBUTION OF THE CPU-LOAD:

LOAD IN PERCENT	0-10	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
DISTRIBUTION IN PERCENT	45.5	9.1	0.0	0.0	9.1	0.0	9.1	0.0	0.0	27.3

AVERAGE LOAD: 39.70 PERCENT

THE DISTRIBUTION OF CONSECUTIVE 90-100 PERCENT LOAD:

NUMBER OF CONSECUTIVE LOADS	1	2	3	4	5	6	7	8	9	10-
DISTRIBUTION IN PERCENT	9.1	18.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

THE TOTAL MEASURINGTIME WAS 1.0 MINUTES WITH 3.00 SECONDS BETWEEN THE READINGS OF THE IDLE-LOOPCOUNTER.

6.5 Infogning i befintligt system

De fem programmen katalogiseras på en discfil som separata tasks med eget resursblock där följande enheter används:

Enhet nummer 3 i FORTRANprogrammen tilldelas LP
Enhet nummer 1 i FORTRANprogrammen tilldelas TY

Vid länkningen skall global COMMON-areor definieras och detta görs med följande fem styrkort inlagda efter EXE EDIT:

```
GCOM AA,(absolutadress),10
GCOM BB,(absolutadress),26
GCOM CC,(absolutadress),400   *)
GCOM DD,(absolutadress),3
GCOM EE,(absolutadress),2
```

*) Observera det som sägs på sidan 6.5 - 6.6 om ändring av buffertens storlek.

De olika programmen använder var och en nedan angivna COMMON - block.

IDLELO	-	-	-	DD	-
CALIBR	-	-	-	DD	-
READO	-	-	CC	DD	EE
CALCUL	AA	BB	CC	-	EE
SUPERV	AA	BB	-	-	EE

Vidare bör kontrolleras att IDLELO respektive READO verkligen hamnar på lägsta respektive högsta prioritet av alla TASKS.

6.6 Anpassning till annat system.

Mätpaketet bör kunna anpassas till de flesta realtidssystem. Några villkor som måste uppfyllas är sammanfattade i de följande sex punkterna:

1. Systemet måste kunna använda program skrivna i FORTRAN.
2. Ett program skall kunna startas upp av ett annat.
3. Program måste kunna fördröjas en viss variabel tid för att därefter reaktiveras.
4. Massminnesresidenta program skall kunna göras kärnminnesresidenta genom programinstruktioner.
5. Tillgång till enhet får operatörskommunikation nödvändig.
6. Uppdelning av program på olika prioritetsnivåer måste vara möjlig.

De konkreta ändringar som krävs är i första hand utbyte av vissa MODCOMPberoende subrutinanrop. Dessa är:

CALL ESTABL(a,b,c,d) som gör program a kärnminnesresident med prioriteten b.
CALL DEESTA(a,b) som gör program a massminnesresident.
CALL TSKDEL(a,b,c,d) som fördröjer programmet där instrutionen finns med a minuter och b tick.
CALL TSKACT(a,b,c,d) som aktiverar program a med prioriteten b.
CALL TSKKIL(a,b,c) som stoppar exekveringen av program a.
CALL TIMDAT(a) som ger datum och klockslag.

---

7. Litteraturreferenser

1. Black: Introduction to on - line computers.
2. Blom: Statistikteori med tillämpningar.
3. Bubenko - Ohlin: Introduktion till operativsystem del I & II.
4. Data processing series: On - line computing systems.
5. Martin: Design of real - time computers.
6. Mieslander: Datorer i reglersystem.
7. Modcompmanualer