JÄMFÖRANDE ANALYS AV DATAMASKINER FÖR PROCESSREGLERING

TOMY ANDOFF o BENGT BODIN

Rapport RE - 25 jan. 1968

JÄMFÖRANDE ANALYS AV DATAMASKINER FÖR PROCESSREGLERING

Examensarbete i regleringsteknik vid Lunds Tekniska Högskola höstterminen 1967

Tomy Andoff

Bengt Bodin

JÄMFÖRANDE ANALYS AV DATAMASKINER FÖR PROCESSREGLERING

Presentation

De uppgifter som förelades oss i detta examensarbete var följande:

1) Att undersöka de krav som ställs på en datamaskin, som skall användas för processreglering
2) Att jämföra prestanda för några olika processdatamaskiner som finns tillgängliga på marknaden
3) Att göra en nyanserad bedömning av dessa maskiner

Som ett resultat av vårt arbete anser vi att den första uppgiften är löst genom den listning av processegenskaper och arbetsuppgifter för processdatamaskiner vi gjort i kapitlen IV och V. Prestandajämförelsen för olika maskiner redovisas i kapitel VIII och bilagorna 1 - 5.

Att göra en nyanserad bedömning av ett maskinsystem utan att ha ett specifikt, väldefinierat processystem som utgångspunkt, anser vi vara mycket svårt, då varje processystem är olika alla andra och därför ställer helt olika krav på de enskilda maskinegenskaperna. (Härtill kommer också det faktum att en praktisk erfarenhet av datamaskiner är en ovärderlig hjälp för bedömning av kraven; denna saknar vi tyvärr.)

Vår förhoppning är att den i kapitel VIII gjorda analysen, där vi har beskrivit de olika lösningar, som finns till de speciella egenskaper förknippade med processdatamaskiner, kan vara till vägledning vid utvärderingar av maskinsystem.

För att begränsa omfånget på vår redogörelse har vi i kapitel VIII sett oss nödsakade att använda en mer avancerad datateknisk terminologi. Läsarens förståelse av texten ökas därför om han har en viss erfarenhet av datateknisk litteratur.

Valet av de maskiner som studerats i vår analys är tämligen godtyckligt. Vi har medtagit maskiner med olika organisation, ordlängd etc. Trots att fullständig dokumentation ännu saknas har vi tagit med LM Erikssons maskin UAC 1601, eftersom den f.n. är den enda svenska processdatamaskinen i marknaden.

Nedan följer en översikt av vårt arbete:

Inledningsvis konstaterar vi att det råder en väldig terminologiförbistring på datamaskinområdet. I kapitel I ger vi en allmän orientering om vilka användningsområden det finns för dagens datamaskiner, för att sedan i kapitel II beskriva olika sätt att utnyttja en datamaskin till processreglering. Vi presenterar i kapitel III en marknadsbild av installerade processdatamaskiner; en tabell över de största leverantörerna inom olika brancher redovisas också. Att det inte går att ge en entydig definition av en processdatamaskin konstaterar vi i kapitel IV. Vi har där emellertid med processkarakteristika som grund beskrivit olika maskinfunktioner som måste finnas. I kapitel V konstaterar vi att de krav man skall ställa på en datamaskin alltid beror på den speciella process den skall reglera. Det gäller alltså att ha så stor kännedom om den ifrågavarande processen som möjligt, för att på så sätt kunna ställa bättre specificerade krav på datamaskinsystemet. Efter systematisering av de arbetsuppgifter som en datamaskin kan utföra i en process ställer vi upp de faktorer och egenskaper man skall bedöma vid en kravanalys av olika maskiner för en viss process. I kapitel VI får vi betona att en bedömning av programvaran hos ett maskinsystem är lika viktig som själva maskinegenskaperna. Vi talar här om programvarukrav, presenterar monitorsystem, olika programspråk, speciella språk för sekvensstyrning och utvecklingstendenser på området.

Några generella metoder för bedömning av processdatamaskinsystems effektivitet behandlar vi i kapitel VII. Vi beskriver två stycken instruktionsblandningar (mixar) (se bil. 3), men vi varnar samtidigt för en alltför ensidig tilltro till dessa. Vid konkreta tillämpningar är ju programvaran minst lika viktig som maskinvaran, speciellt om man beaktar kostnaden för systemarbetet som ofta är av samma storlek som maskinvarupriset.

Vidare ställer vi i punktform upp allmänna viktiga frågor som bör studeras vid effektivitetsbedömningen. Slutligen tar vi upp den alltmer använda simuleringstekniken.

De egenskaper hos processdatamaskiner, som vi mer ingående beskriver i kapitel VIII är brytsignalsystem, minnesskydd, A/D- och D/A-omvandlare och programvarupaket. Övriga studerade egenskaper finns uppställda schematiskt i bil. 1, 2 och 4.

Innehållsförteckning

Förord

Då man börjar en studie av processdatamaskiner, konstaterar man genast att det har byggts ett väldigt Babelstorn på detta område. För att lösa detta terminalogiproblem har vi försökt använda svenska termer i så stor utsträckning som möjligt (t.ex. hardware = maskinvara; software = = programvara). För att läsaren skall förstå vårt språk rekommenderas studium av vår ordlista innan genomläsningen av uppsatsen börjas.

I vårt arbete har vi valt att undersöka relativt stora maskinsystem. Vi vill emellertid här göra läsaren uppmärksam på att det nu finns mindre och därmed billigare maskinsystem i marknaden.[1)]

I DATABEHANDLINGSSYSTEMENS ANVÄNDNINGSOMRÅDEN

Då man skall klassificera databehandlingssystem (DBS - def. se ordlistan) finns det flera indelningsgrunder att utgå från. Det vanligaste och enklaste sättet att genomföra uppdelningen är att studera DBS's användningsområden.

Matematiska och tekniska beräkningar är den typ av arbeten för vilka datamaskinsystemen (DMS - def. se ordlistan) ursprungligen konstruerades. Det utgör fortfarande ett av de ekonomiskt sett viktigaste områdena.

Administrativa databehandlingen är det nu volymmässigt dominerande verksamhetsfältet.

Statistiskt arbete är ett lämpligt område för DMS, antingen som separat arbetsuppgift eller som "bisyssla".

Planeringsarbeten av olika slag, t.ex. optimeringsstrategier vid produktplanering inom industrin, nätverksplanering m.m.

Informationssökning (eng. information retrieval) innebär att i register söka upp viss specificerad information. Detta är ett område där DBS har

---

1) T.ex. Interdatas Model 3, Scientific Control Corp. Model 650, Business Information Technology Inc. BIT 480, Data Machines's 620-I och PDP 8 S. /Ref. nr 5/ (se ref.lista)

en jätteuppgift framför sig. Ett omfattande katalogisteringsarbete har startats bl.a. inom medicin- och kemiområdena. /Ref. nr 12 och 9/

Översättning av text från ett språk till ett annat är ett problem som det arbetas med på många håll i världen. /Ref. nr 8/

Språkforskning t.ex. frekvensanalys av text för spårandet av författare.

Vid sidan av beräkningsuppgifterna och den rena administrativa databehandlingen spelar också processregleringen en växande roll.

II DATAMASKINSYSTEM FÖR PROCESSREGLERING

Det finns i princip tre olika sätt att i en process utnyttja ett DMS: /Ref. nr 9/

1. Operators Guide Control (operationsledning)
   DMS är här kopplat till processen men ingriper ej i den.

   Olika nivåer av komplexibilitet finns här: T.ex. samlas information och presenteras på läsbar digital form för att utgöra operations-, kontroll- eller statistikunderlag. Nästa steg är att insamlade data analyseras, avlästa värden jämförs med föregående data och resultaten utskrivs i lämplig form för processoperatören och övriga företagsfunktioner. Om man vid analysen även tar hänsyn till redan lagrade styrplaner kan man erhålla direkta styrorder till operatören, som i sin tur manuellt ställer in processens regulatorer.

   När det gäller informationens relevans har man vissa problem. Det finns i alla processer många informationer som är svåra att direkt mäta; instrument saknas eller är för dyrbara, mättekniken ännu ej utvecklad etc. Här får man med hjälp av övrig information göra nödvändiga uppskattningar av saknade värden.

2. Set Point Control (börvärdesstyrning)
   DMS är här kopplat till processen via regulatorer. Det känner av mätsignalerna från givarna och beräknar nya börvärden med hjälp av programmerade styrlagar för processen. Regulatorerna ställes sedan via signaler från DMS in på dessa värden direkt.

Fördelen med detta system är att man kan klara mera komplicerade och snabbare förlopp än ovan.

3. Direct Digital Control (DDC) (digital direktstyrning)
DMS ingriper här direkt i processen utan hjälp av regulatorer som mellanhänder. DDC-systemet kan vara en optimal lösning av regleringen. Vid DDC ställs mycket stora krav på processdatamaskinsystemets (PDMS) driftsäkerhet. I ett set-point-system kan operatören fortfarande ställa in regulatorerna om DMS skulle falla ifrån. I DDC-systemet måste man ha en betryggande back-up (reservmöjlighet) då man inte annat än genom DMS kan gå in i processen. Då ett haveri eller driftstopp kan kosta företaget lika mycket som DMS, förstår man att stor vikt måste läggas på driftsäkerhet och back-up-möjligheter i detta fall. Man talar nu om att, som en optimal lösning på säkerhets-kostnads-problemet köra med en dubbel uppsättning DMS /ref. nr 3/. Här bör dock påpekas att DMS har en felfrekvens som är många gånger mindre än de givare och reglerdon som ofta är anslutna till DMS.

III MARKNADSBILD

I och med övergången till den tredje generationens DMS har antalet PDMS ökat kraftigt. År 1962 fanns det ca 200 st i hela världen mot de f.n. år 1967 ca 1600 st installerade eller beställda /Ref. nr 4/. Merparten av dessa är dock av relativt enkel typ (Operators Guide Control).

Det totala antalet PDMS med DDC är f.n. ett 40-tal. De i Sverige installerade eller beställda PDMS är ca 15 st /Ref. nr 9/. De sex största leverantörerna är (enl. Control Engineering's sammanställning i mars 1967) /Ref. nr 4/

General Electric	216 st
IBM	133 st
Scientific Data Systems	127 st
Westinghouse	118 st
GEC + CAE[1)]	109 st
Honeywell + 3 C	108 st

[1)] General Electric Co (ej GE i USA) Cie Européenne d'Automatisme Electronique

Vid uppdelning på branscher fås följande fördelning av leverantörer:

Petrolium, kemi, papper, cement	
IBM	68 st
General Electric	64 st
Bunker-Ramo	49 st
Järn- och stålmetaller	
Westinghouse	51 st
General Electric	46 st
Honeywell + 3 C	16 st
Elkraftgenerering och -distribution	
General Electric	72 st
Westinghouse	38 st
Bailey Meter	33 st
Övriga områden	
Scientific Data Systems	116 st
GEC + CAE	65 st
Honeywell + 3 C	53 st

Den allmänna utvecklingen leder nu mot en eftersträvan att totalintegrera DMS med företaget. Man söker här sammanlänka PDMS med det administrativa DBS. Detta kan leda till ett flertal olika datamaskiner på olika nivåer.

Det mest avancerade systemet i Sverige idag - IBM´s i Billerud - innehåller bl.a. ca 18 set-point loopar och 14 DDC loopar /Ref. nr 13/.

IV KARAKTERISTIKA FÖR ETT PROCESSDATAMASKINSYSTEM

Att ge en entydig definition av ett PDMS är en omöjlig uppgift. Vi kan emellertid försöka ange de mest karakteristiska egenskaperna som finns hos dagens PDMS. För att komma dit ställer vi först frågan: Vad sker normalt i en process? Vi finner bl.a. följande /Ref. nr 6/:

Flera funktioner utföres simultant. Funktionerna kan vara

- kontinuerliga - t.ex. analog avläsning, styrning
- periodiska - t.ex. timlogging (avläsning med bestämda tidsintervaller av ett mätvärde)
- slumpmässiga - t.ex. alarmering vid fel

Olika funktioner har olika prioritet.
Kösituationer inträffar.
Inbördes förhållanden mellan olika funktioner ändras, t.ex. ändring av drifttillstånd.
Klimatet ändras.

Detta medför att PDMS

1. Tar emot och levererar flera olika slag av data samtidigt (både analoga och digitala signaler)
2. Arbetar kontinuerligt (inget driftstopp får förekomma)
3. Lagrar vissa indata, som sedan bearbetas satsvis (batch-processing)
4. Har prioritets- och köordnande funktioner
5. Är flexibelt i maskin- och programvara
6. Är okänsligt dels för variationer i luftfuktighet och temperatur och dels för t.ex. elektriska störningar från omgivningen

V KRAV PÅ PROCESSDATAMASKINSYSTEM

De krav man kan ställa på ett PDMS är många och mycket varierande, beroende på vilken typ av process det skall arbeta i. Då även inom samma branch skillnaderna mellan de olika industriprocesserna är mycket stora, medför detta att varje anpassning av PDMS till ett speciellt förlopp fordrar lösningar förknippade endast med ifrågavarande process. Vissa försök till standardisering av industrienheter sker just nu och det talas även om att bygga industrier med integrerat PDMS. Innan vi försöker analysera de krav som ställs på ett PDMS, studerar vi de uppgifter det vanligen utför. Här följer en tänkbar uppdelning efter arbetsuppgifternas förekomstfrekvens/Ref. nr 10/:

Nivå 1 (högsta frekvens)

a) Utskrift på skrivmaskin, remstans etc.
b) Avläsning och konvertering av analoga och digitala signaler, test av överskridna gränser, indikering av onormala tillstånd, trendövervakning
c) Utmatning av analoga och digitala signaler
d) Överföring av data och program till och från minne
e) DDC-styrning
f) Feed-forward styrning (framkoppling)

Nivå 2

a) Rapportering, beräkning av prestanda och produktion
b) Kommunikation med operatör
c) Beräkning av optimala börvärden
d) Sekvensstyrning vid ändring av drifttillstånd
e) Ändring av styrparametrar (adaptiv styrning)

Nivå 3 (lägsta frekvensen)

a) Uppdatering av parametrar i de matematiska modeller som möjliggör t.ex. beräkning av optimala börvärden och feed-forward styrning på nivå 2
b) Produktionsplanering
c) Ändring av program och programstruktur
d) Icke processbundna bearbetningar

Denna allmänna uppställning är inte på något sätt tillräcklig för att kunna precisera kraven på ett PDMS. Härför kräves en noggrann analys av den aktuella processen. Följande faktorer anser vi vara av stor vikt:

1. Systemets allmänna utformning
   d.v.s. balans mellan maskin- och programvara, möjlighet till expansion och flexibilitet i maskin- och programvara och maskin- och programvarans anpassning till processen
2. Inre och yttre minnen
   a) storlek
   b) överföringstider
   c) accesstider
   d) cykeltid
   e) ordlängd
3. Direkt access till minnet (cycle stealing)
4. Ordlängdens anpassning efter minnesutrymmet och assemblerns effektivitet
5. Brytsignalsystemets utformning
6. Minnesskyddets utformning
7. Multiplikation/division (realisering i maskinvara eller programvara?)
8. Instruktionslistans utformning
9. Tillgänglig programvara (standardprogram)

10. Möjlighet till multiprogrammering
11. Adresseringsmoder (relativ, indirekt och indexerad adressering)
12. Driftsäkerhet
13. Service och reparation
14. Leveranstid och installation } vid total bedömning
15. Totala kostnader

I den efterföljande analysen (kap. VIII) kommer vi i detalj att presentera olika fabrikanters lösningar av problem associerade med de flesta av ovanstående faktorer.

VI PROGRAMVARA FÖR PROCESSDATAMASKINSYSTEM

Kravet på ett välutvecklat programsystem har hittills behandlats rent styvmoderligt av såväl köpare som tillverkare av DBS. Men kostnaderna, tillförlitligheten, önskan om flexibilitet och bristen på programmerare har under de senaste fem åren gjort styvmodern till arvtant. De olika fabrikanterna lägger nu ner stor omsorg på att standardisera och bygga upp omfattande programsystem i modulform. Tyvärr existerar det här ej någon internationell standard, varför språkförbistringen på området är mycket svår.

För att ta fram dessa nya program använder man moderna produktionsprinciper, t.ex. operationsanalytiska. Från att möjligen ha varit användbara hjälpmedel är programsystemen nu lika betydande som de elektroniska enheterna. Då vi i nedanstående studie av tillverkarnas manualer tyvärr endast har tid och kunskaper att behandla programvaran ytligt, skall vi redan här ge en allmän överblick över de programsystem som f.n. erbjuds kunderna.

1 Programvarukrav

Den på sid. 5 /Ref. nr 10/ gjorda arbetsrutinuppdelningen för PDMS ger en antydan om en möjlig standardiseringsgrad. Ju längre från processen man kommer, d.v.s. på ju högre nivå, desto mindre är möjligheten att använda standardlösningar.

Uppdelningen ger också en bild av den önskvärda hierarkin för de program som skall ombesörja funktionerna. En sträng rangordning mellan olika

program, begränsning av programfunktionerna samt väldefinierade I/O-data underlättar programmering, inkörning och modifiering.

Vi kan också konstatera att komplexibiliteten och storleken av de olika programmen ökar med nivån, men då frekvensen avtar blir ofta räknevolymen ungefär lika fördelad mellan nivåerna.

2 Monitorsystem

Den största skillnaden mellan teknisk eller administrativ programmering och programmering för processtyrning är den svårighet som tidsdimensionen medför, De avbrottsignaler som kommer in, dels från processen, dels från normala perifera organ, gör t.ex. att program, som körda ensamma fungerar perfekt, vid samtidig körning ej fungerar eller, ännu farligare, ibland ej fungerar. Här kommer således leverantörerna idag programmerarna till hjälp (med reelltidsproblemen).

En stor uppsättning standardprogram inkluderas idag i leveransen av PDMS. Delar av dessa programsystem brukar kallas monitorprogram eller -system. På detta område råder mycket stor begreppsförvirring och vi hänvisar här till resp. leverantörers manualer.

De standardprogram och rutiner som kan ingå i sådana system framgår av tabellen nedan.

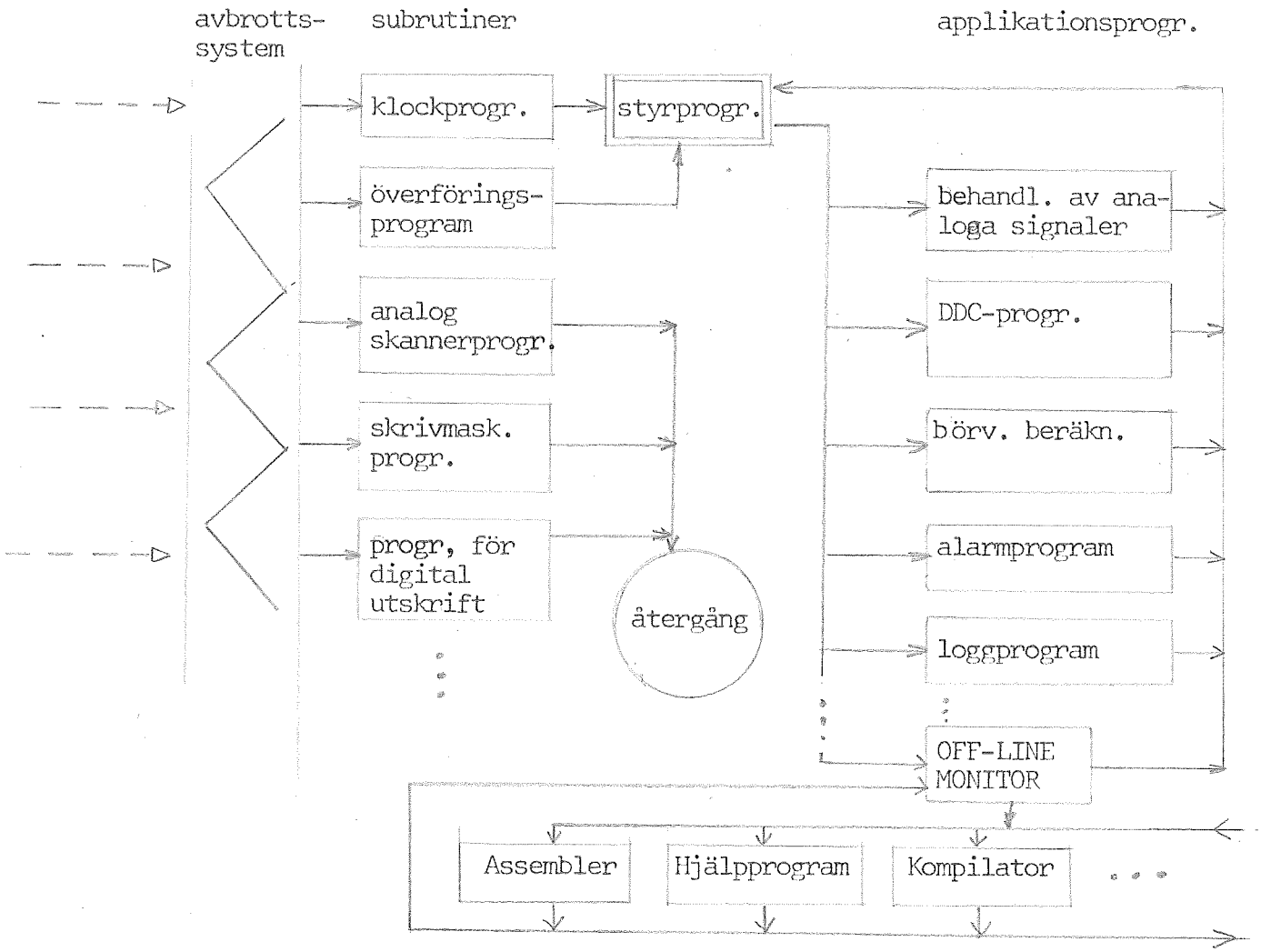
I figuren visas ett allmänt exempel på hur dessa standardprogram och rutiner kan samarbeta med applikationsprogrammen.

Tabell Standardprogram och rutiner som kan ingå i ett monitorsystem

Styrprogram (def. se nedan)
Assembler
Kompilator
Program för analog I/O matning
Program för digital I/O matning
Program för I/O matning till perifera organ inkl. plotter
Program för övervakning av perifera organ
Rutiner för start, fördröjning och stopp av program, registerhantering, avbrottsrutiner

Rutiner för överföring från och till yttre minnen
Matematiska rutiner, enkel - dubbelprecision
Hjälprutiner för programtest
Program för skalning av analoga insignaler
Program för redigering av utskrifter

Fig. Exempel på programstruktur för PDMS /Ref. nr 10/



Styrprogrammet utgör den viktigaste delen av monitorsystemet och har till uppgift att med kännedom om verklig tid och med hänsyn till inbördes prioritet starta applikationsprogram vid givna tidpunkter eller på begäran. Det måste också tilldela programmen minnesutrymme och överföra dem från yttre minnen om de är lagrade där.

Styrprogrammet har för detta ändamål tabeller över hur olika minnen utnyttjas, programmens lägen och längder, information om återstartspunkter för avbrutna program o.s.v. Om ett program vid inmatning i kärnminnet

inte får plats där, sköter styrprogrammet om utskyffling av något annat program, som under tiden lagras i yttre minnet.

Under den tid som behövs till överföringen, återgår programkontrollen till det avbrutna programmet. Enl. figuren aktiveras styrprogrammet via avbrottssystemet, förutom vid klockpuls även efter varje avslutad överföring.

De funktioner som utföres under normal drift av ett PDMS brukar ej utnyttja den totala kapaciteten hos CPU, då ju in- och utmatningen av data tar mycket längre tid i anspråk än behandlingen i CPU. Den outnyttjade lediga tiden brukar då användas till icke processbundna bearbetningar. Man brukar därför ha ett off-line monitor-system, se fig. ovan. Exempel på arbetsuppgifter som här utföres är kompilering och inkörning av nya applikationsprogram. Denna teknik fordrar att on-line programmen kan avskärmas med hjälp av minnesskydd.

Perifera organ, t.ex. konsolskrivmaskiner, analoga och digitala I/O enheter, dirigeras av avbrottsinitierade subrutiner (se t. vänster i fig.). Dessa arbetar med datatabeller i kärnminnet, i vilka informationen lagras eller varifrån den utmatas, samt med tabeller med styrinformationer, d.v.s. adresser, skalfaktorer o.s.v.

3 Programspråk /Ref. nr 6 och 10/

En process styrs inte bara med hjälp av monitorsystemets standardrutiner, utan det fordras också vissa, direkt för den aktuella processen skrivna program, applikationsprogram. För att detta skall ske på ett enkelt sätt behövs effektiva programspråk.

Som bekant innebär övergången från maskinkodsprogrammering till symboliska språk (assemblerspråk) ett väsentligt framsteg med avseende på flexibilitet, minskad felfrekvens och kortare programmeringstid utan att maskinsystemet utnyttjas sämre. Idag har varje DBS ett väl utvecklat assemblerspråk som standard, med möjlighet till makroinstruktioner. Dessa utläggs vanligen av assemblern som hopp till standardsubrutiner.

Vissa svagheter finns med symbolkoder. De fordrar stor detaljnoggrannhet som medför lång programmerings- och testtid. Kommentarer krävs för

förståelsen och koden är också bunden till en viss maskintyp.

Inom konventionell databehandlingsteknik har som bekant högre språk använts i snart 10 år för att kringgå dessa svårigheter, t.ex. de problemorienterade språken FORTRAN och ALGOL inom teknisk databehandling och COBOL inom administration. Genom att bl.a. mängder av småfel undvikes med dessa språk kan programmeringstiden avsevärt reduceras. Nackdelar med de färdiga programmen är att de blir längre än de problemorienterade.

Då man får bättre metoder för att bygga effektiva översättningsprogram, minskade minneskostnader och snabbare maskiner, får man i framtiden räkna med större användning av problemorienterade språk även för processtyrning med PDMS.

Dagens läge kännetecknas dock av att assemblerspråk används i stor omfattning. De är praktiskt taget allenarådande för framställning av monitorsystem och övriga program på lägsta nivån enligt det föregående.

4 Speciella språk

För vissa tillämpningar,såsom sekvensstyrning (t.ex. uppstartning av kraftverk) och DDC,har speciella programsystem utvecklats.

För sekvensstyrning används ofta en teknik med villkorstabeller (beslutstabeller). Ordningsföljden mellan konsekutiva åtgärder bestäms av en mängd boolska tillståndsvarabler. Utförandet av en viss funktion sker endast då ett visst funktionsvärde (bitmönster) är förhanden. Fördelen med detta system är att programmeringen av processekvensen sker genom ifyllandet av standardiserade tabellblanketter. Modifieringar av funktionssättet sker genom ändring av vissa ord i maskinminnet. Exempel på sådana språk: GE´s TASC och AEI´s Overseer.

Också för DDC-styrning har speciella språk eller system utvecklats som ger en smidig definiering av styrstrategier, samplingshastigheter o.s.v. Det resulterande programmet innehåller ofta ett eget styrprogram, som är underordnat det allmänna styrprogrammet. Här uppstår ofta svåra problem vid anpassningen av DDC-program till det överordnade systemet.

5 Utvecklingstendenser

Av det ovanstående framgår att det idag finns i princip fyra olika typer av programsystem och språk:

1. Monitor med styrprogram och standardrutiner
2. Speciella processorienterade system för t.ex. sekvensstyrning och DDC
3. Assemblerspråk
4. Kompilatorspråk

Man kan konstatera att de två första typerna av system är skrivna med hjälp av assemblerspråket. De blir därför dyra att framställa och svåra att ändra. Visserligen sammanställs ett monitorsystem av lämpliga standardrutiner till en för det aktuella systemet avpassad konfiguration, men delarna i sig är låsta. Detta innebär att monitorsystemet ej passar så bra för vissa speciella tillämpningar, t.ex. enbart för DDC eller för små system.

Lösningen på problemet är ett programmeringsspråk speciellt konstruerat för relltidsprogrammering med en vokabulär som kan beskriva hur reell-tidsmaskinvara skall arbeta. Endast härigenom kan det användas till att skriva styrprogram och monitorprogram, vilket är ett väsentligt syfte med detta språk. Motsatsförhållandet generalitet - anpassning till maskinvaran utgör svårigheten.

VII DISKUSSION AV GENERELLA METODER FÖR BESTÄMNING AV PROCESSDATAMASKIN-SYSTEMETS EFFEKTIVITET

Av vårt hittills förda resonemang framgår att ett DBS´s kapacitet ej endast är beroende av PDMS´s kapacitet, utan av den samlade effekten av PDMS och programvaran, applicerat på ett bestämt processystem, där man speciellt tar hänsyn till systemets beteende och informationskrav.

1 Mixar

När man på ett tidigt stadium, då kännedomen om processen är liten, vill göra valet av en viss bestämd maskinell utrustning och skall ha någon beslutsgrund för valet brukar i denna studie ingå jämförelse mellan de olika maskintypernas interna hastigheter. I brist på närmare kännedom

om den bearbetning som skall utföras i framtiden, tillgriper man en för applikationsområdet representativ instruktionsblandning (mix).

För matematiska och fysikaliska beräkningar har en s.k. "vetenskaplig instruktionsblandning" (scientific mix) angivits av Gibson. Denna blandning anger frekvenser för olika instruktionstyper hos de här använda programmen. För processer av administrativ karaktär vid reelltidssystem har Raichelsen och Collins komponerat motsvarande reelltids mix. För vidare studium av dessa två mixer, se bilaga nr 3 /Ref. nr 1/.

Detta sätt att jämföra CPU's interna kapacitet med hjälp av blandningar anser vi alltför överskattad. Ungefär samma resultat nås på ett enklare sätt genom jämförelse mellan CPU's olika räknetider, se tabellen i bilaga nr 4. Man kan i mixerna t.ex. ej ta hänsyn till vissa tidsbesparande instruktioner (t.ex. för bearbetning av brytsignaler m.m.). Samspelet mellan bearbetning i CPU och datatransporter belyses ej. Datatransporttiderna är ofta av mycket större betydelse.

Även om man gör beräkningar av interna räknehastigheter, datatransporthastigheter och i mer ambitiösa fall tidsberäkningar av några ofta förekommande produktionskörningar, leder detta till en underdimensionering av maskinsystemet. Anledningen är att man då bortser från systemets beteende under drift och de spilltider som då oundvikligen uppkommer.

2 Bedömning av processystem

Tidsbehovet för testning och inkörning av nyutvecklade program får heller inte underskattas. Vid processystem, då behovet varierar slumpmässigt med tiden, är analytiska metoder för beräkningar av konfiguration och kapacitet antingen helt överslagsmässiga eller svårhanterliga p.g.a. dess komplexibilitet med mängder av simultanitet, prioritetsfrågor och olika köbildningar. Här får man bl.a. försöka studera följande frågor /Ref. nr 1/:

1. Vilken utnyttjandegrad får systemets olika facititeter av den aktuella arbetsstrukturen (job-strukturen) och belastningen?
2. Hur kan man beräkna systemets kapacitet? Detta betecknas i litteraturen som "throughput"-kapacitetsnivån och uttrycks vanligen som antalet behandlade arbeten/tidsenhet.

3. Hur lång tid tar det för de olika arbetena att gå genom systemet? Hur långa väntetider bör man räkna med?
4. Hur beter sig systemet för olika belastningar upp till den teoretiskt maximala kapacitetsnivån? Hur påverkas de enskilda faciliteternas utnyttjandegrad och de olika väntetiderna i systemet av variationerna i arbetsbelastningen?
5. Hur påverkas kapacitetsnivå, väntetider m.m. av bestämda ändringar i systemets konfiguration (ökat primärminne, kanalenheter m.m.) eller av ändringar i planeringsalgoritmerna?
6. Vilka planerings-algoritmer och -parametrar bör tillämpas för att få optimalt nyttovärde av systemet vid given konfiguration?

Då det tyvärr ej är så enkelt att det existerar generella metoder, som ger svar på ovanstående frågor, får man bygga modeller med vars hjälp man simulerar processystemet.

3 Simulering

Modellens grad av komplexibilitet och likhet med det verkliga systemet är ett resultat av den svåra avvägningen: tid + kostnad + god approximation av verkligheten kontra enkelhet för att bli matematiskt och simuleringstekniskt analyserbart.

Simulering är den metod för analys som blir allt vanligare. Man kan härigenom upptäcka flaskhalsar i systemet och mäta olika enheters utnyttjandegrad och de olika arbetenas genomloppstid som funktion av olika driftsstrategier. Flera s.k. simuleringsspråk[1)] för DBS har utvecklats. Dessa lättar programmeringsbördan och beskriver på ett smidigt sätt komplexa system.

[1)] Till de mest kända räknas GPSS (av Gordon), SIMSCRIPT (Markowitz Hausner och Karr) och SIMULA (Dahl och Nygaard) som är byggt på ALGOL

VIII JÄMFÖRANDE ANALYS AV NEDANSTÅENDE MASKINER MED AVSEENDE PÅ SIGNIFIKANTA EGENSKAPER GENOM STUDIER AV RESPEKTIVE FABRIKANTERS MANUALER

CDC 1700
GE/PAC 4000
IBM 1800
PDP 8
PDP 9
Sigma 2 (GEC S 2)
UAC 1601

Vi kommer att redovisa en studie av manualerna för några av de PDMS som f.n. finns på marknaden. Vid ovanstående val av maskiner har vi försökt att få med sådana som för vår analys har intressanta lösningar på några av de problem som är förknippade med de på sid.6 och 7 diskuterade faktorerna. Exempelvis varierar ordlängden hos maskinerna från 12 till 24 bitar. Vi har i fortsättningen varit tvungna att använda resp. fabrikanters terminologi; att här försöka göra en översättning av använda begrepp till svenska termer anser vi vara omöjligt och onödigt.

Det är vår förhoppning att läsaren genom denna redovisning skall få en känsla för vilka faktorer som kan vara mer eller mindre kritiska. För att inte segla ut på oändlighetens ocean har vi helt avstått från att medtaga den ekonomiska faktorn i vårt arbete.

En sammanställning i tabellform över systemfunktioner för maskinerna CDC 1700, GE/PAC 4000, IBM 1800, PDP 8, PDP 9, GEC:S2, UAC 1601, SIEMENS SYSTEM 300, ARCH 102, ARCH 9000 och ARCH 2020 medföljer som bilaga (bil. 2).

Bilaga 1 visar registeruppsättningen för de sju i analysen studerade maskinerna. Dessa maskiners systemkonfigurationer återfinnes i bil. 5.

1 Brytsignalsystem

Brytsignalsystemets uppgift är att generera prioritetsavbrott med ledning av utifrån kommande stimuli. Detta möjliggör att maskinen kan arbeta på time-shared basis. (Detta får nästan betraktas som ett villkor för att maskinen skall kunna utnyttjas effektivt.) Orsaken till ovanstående är att det tar mycket längre tid att mata data in och ut från

processorn än det tar att utföra aritmetiska och logiska operationer. Time-sharing gör det möjligt för andra program att utnyttja väntetider till att få sina operationer utförda. Vid utformningen av brytsignal-systemet har man att ta hänsyn till bl.a. följande faktorer.

Skall prioritetsavbrotten göras med hjälp av maskinvara eller programvara och hur många prioritetsnivåer skall finnas? Maskinvaru-metoden är den snabbaste, men tyvärr även den mest kostsamma. Här får alltså applikationen bestämma tillvägagångssättet.

Antalet prioritetsnivåer är mycket varierande från maskin till maskin och någon generell regel för hur många nivåer som bör finnas går svårligen att uppställa. Nedan följer en beskrivning av brytsystemets uppbyggnad hos några olika maskiner.

CDC 1700 /Ref. nr 15/

CDC 1700 har max. 16 st avbrottsnivåer. Grundenheten 1704 har endast två interna avbrott (paritetsfel och programskydd) på en nivå och ett avbrott (från Teletypewriter) på en annan nivå. Komplettering med 1705 (Interrupt Data Channel) ger ytterligare 15 externa avbrott. Ett av dessa är eller-anslutna till Teletypewriterns avbrott.

Avbrottsystemet består av bestämda interrupt trap locations (utrymme i minnet där samtliga registers innehåll lagras vid ändring av prioritetsnivå) och av ett 16-bitars maskregister. Innehållet i maskregistret bestämmer prioriteten på det speciella avbrottstillståndet. Varje avbrottslinje (= linje genom vilken kan komma begäran om prioritetsavbrott) är alltså associerad med ett visst innehåll i maskregistret. Med avbrottslinjen är även associerad ett tal, 0-15, vilket representerar prioriteten på de program som användes vid det aktuella innehållet i maskregistret.

Två villkor måste vara uppfyllda innan en avbrottsignal kan uppträda:

1) Maskregisterbiten för detta avbrott måste sättas set av en speciell instruktion (Inter-Register-Instruction)
2) Avbrottsystemet måste aktiveras med en speciell order (Enable Interrupt). Nedan följer en redogörelse för en av många möjliga metoder att använda avbrottsystemet.

När en avbrottsignal uppträder lagrar maskinen (automatiskt) undan återhoppsadressen i de 15 minst signifikanta positionerna i en minnesadress som är reserverad för detta avbrottstillstånd. Bit 16 är set eller cleared med hänsyn till overflow - indikatorns innehåll. Avbrottsystemet inhiberas sedan och kontrollen överflyttas till en annan adress som specificeras av avbrottstillståndet. Programmet lagrar därefter alla register, inkl. maskregistret i minnespositioner som är reserverade för detta avbrottstillstånd, laddar maskregistret (och även övriga register) med det innehåll som skall användas för det nya tillståndet och upphäver slutligen blockeringen av avbrottsystemet.

Vid utträde från avbrottstillståndet inhiberar programmet avbrott och lagrar återigen samtliga register, som tidigare var placerade i reserverade minnespositioner. Programmet utför sedan instruktionen Exit Interrupt. Denna instruktion innebär att återhoppsadressen (return address) läses från den position där den tidigare undanlagrats, overflow-indikatorn sättes med hänsyn till bit 16 och avbrottsystemet aktiveras. Kontrollen överflyttas sedan till återhoppsadressen och programmet fortsätter. Fördelen med detta system är enligt manualerna att det tillåter prioritetsavbrotten att stå under kontroll av programmet, vilket kan tilldela eller ändra prioritet genom att ändra korrespondensen mellan innehållet i maskregistret och prioritetsnivån.

GE/PAC 4000 /Ref. nr 16/

GE/PAC 4000 innehåller i grundutförande 8 st API (Automatic Program Interrupt) med möjlighet till utvidgning till 128 i grupper om åtta.

Avbrottssignalerna är anslutna till flip-flops som detekteras inom GE/PAC. API-modulen är ansluten till GE/PAC-systemet via den aritmetiska enheten. Den totala söktiden för åtta avbrott är ung. 8 us.

Med hjälp av avbrottsmodulen och instruktionen Decrement Memory and Test (DMT) kan API betjäna sådana funktioner som pulsräkning, "accumulation and elapsed line counting".

Följande klassificering av API´s inputs kan uppställas:
Cycles timer pulse input
Pulser (från processor)
Process alarm
Perifer utrustning klar

GE/PAC erbjuder även något som man kallar Dynamic Priority Control. Detta möjliggör för den internt lagrade programlogiken att tillåta eller inhibera antingen individuella API eller grupper om åtta API, med hänsyn till ett "maskord" som innehålles i minnet. Det går dock ej att inhibera alla typer av avbrott. Med hänsyn till detta klassificeras programavbrotten i Inhibitable Interrupts och Non hibitable Interrupts. Exempel på de senare utgör sådana, där signalkällan t.ex. är Cycles Timer Pulses eller Pulse Source.

Låt oss lite närmare betrakta klassen Inhibitable Interrupts. Utförandet av instruktionen IAI (Inhibit Automatic Interrupt) eller SPB (Store Place and Branch) överför maskinen i Program Interrupt inhibited mode. Inhiberade avbrott kommer att fördröjas tills instruktionen PAI (Permit automatic Interrupt) utföres och därmed återför maskinen till Program Interrupt permitted mode. Instruktionerna LDP (Load P from location Y) och LPR (Load P from location Y and Restore) kommer antingen att inhibera eller tillåta avbrott beroende på bit 21 i den refererade minnescellen.

Instruktion SPB är så pass viktig och intressant att den är berättigad närmare presentation. SPB sparar i indexregister 1 huvudprogrammets status (avbrott, test overflow och programräknare), inhiberar avbrott och transfererar programkontrollen till den i operanden anvisade adressen. Återhopp från subrutinen till huvudprogrammet utföres med instruktionerna LDP eller LPR vilka återlagrar sparade status.

Vi övergår nu till att betrakta klassen Noninhibitable Interrupts. Som tidigare nämnts har programmet ingen kontroll över dessa. Ett programavbrott är en "execute" funktion, d.v.s. en instruktion inskjutes i den normala sekvensen

medan programräknaren inhiberas. Adressen på den inskjutna instruktionen genereras av prioritetslogiken från den verksamma positionen i avbrottsregistret. Den inskjutna instruktionen är normalt en av följande fyra:

DMT - Decrement Memory and Test
SPB - Store Place and Branch
NOP - No Operation
BRU - Branch

Normalt användes SPB för Inhibitable Interrupts och DMT för Noninhibitable Interrupts. NOP användes som temporärt hjälpmedel att sätta ett interrupt-driver program ur stånd. BRU användes endast vid inhopp till nödprogram, eftersom det inte tillåter ett återhopp till huvudprogrammet. SPB som kommer ihåg läge och status för huvudprogrammet brukar användas vid hopp till subrutiner som är utformade för betjäning av de begärda avbrotten. DMT brukar användas vid pulsackumulering (räkning) eller system-tidhållning.

IBM 1800
/Ref. nr 17/

IBM 1800 har basic 12 st externa och extended 24 st prioritetsnivåer. För att kunna redogöra för avbrottsystemets funktionssätt kräves en del förklaringar av begrepp som kommer att användas.

Förutom de nämnda 12 prioritetsnivåerna ingår även följande i basic-utförandet: Internal, Trace och CE avbrottsnivåer,

Internal Interrupt inträffar när någon av följande utsagor är uppfyllda

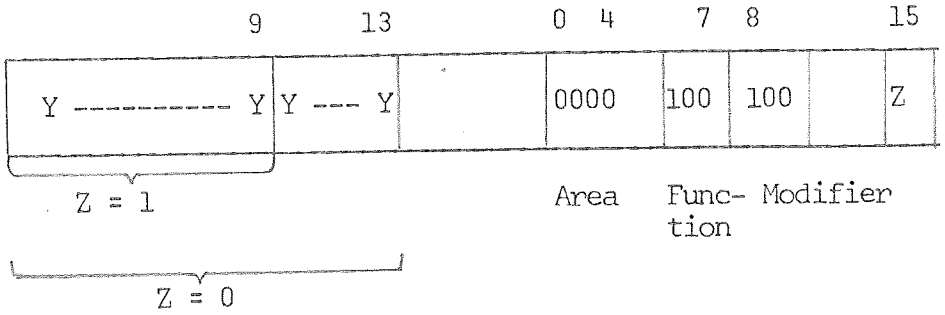
1) En otillåten Op-kod uppträder
2) Ett paritetsfel upptäckes vid överföring mellan Storage Buffer Reg. och minne (och tvärtom)
3) CAR Check (CAR = Channel Address Reg.) indikerar fel

Märk: Internal Interrupt kan ej maskas.

Trace Interrupt äger rum efter varje instruktion om P-C (Processor-Controlers) är programstyrd med konsolens omkopplare i läge Trace.

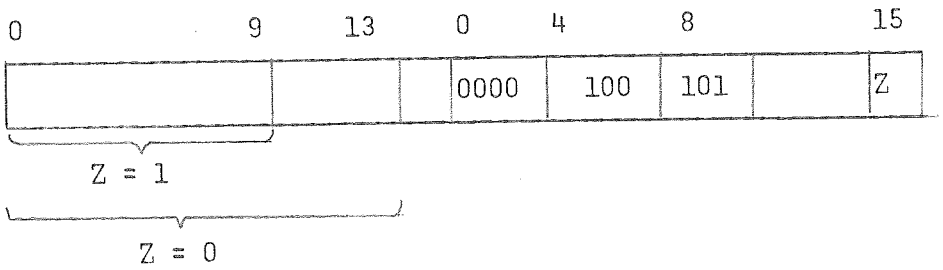
Beträffande CE interrupt hänvisas till manualen.

För maskning och avmaskning av externa avbrottsnivåer finnes ett maskregister. En avbrottsnivå som är maskad kan inte begära betjäning förrän den blivit avmaskad. Execute I/O (XIO) Control-instruktionen användes vid maskning och avmaskning av avbrottsnivåerna 0-13 eller 14-23, beroende på bit 15 (Z) i IOCC. (IOCC står för I/O Control Comand och har följande utseende vid maskininstruktion:



En 1-bit maskar motsvarande avbrottsnivå
En 0-bit avmaskar motsvarande avbrottsnivå

Externa avbrottsnivåer kan programmeras. En XIO Control instruktion användes för att slå på en individuell extern avbrottsnivå inom en av grupperna 0-13 eller 14-23, beroende på bit 15 i IOCC. IOCC har då nedanstående utseende:



I/O organen hos 1800-systemet och några av systemdetaljerna innehåller statusindikatorer. On/off tillståndet för varje statusindikator visar för operationsprogrammet, i vilket tillstånd det organ befinner sig i vilket indikatorn är placerad. De process- och systemindikatorer som är tilldelade avbrottsnivåer initierar begäran om avbrott då de kopplas på.

En XIO Sense Device instruktion, vilken specificerar ett bestämt organ, användes för att i A-registret (Ackumulatorregister) inläsa on/off tillståndet hos varje indikator som är belägen i det speciella organet. Då detta inlästs i A-reigstret betraktas dess innehåll som ett Device Status Word (DSW) eller ett Process Interrupt Status Word (PISW), beroende på huruvida organet ifråga är beläget inom systemet eller i processen.

Förklaring av begreppet Interrupt Level Status Word (ILSW): Liksom PISW och DSW är ILSW i verkligheten inget ord förrän det är inläst i A-registret. Tidigare är ILSW helt enkelt 16 st signalledningar till vilka (var och en av de 16 ledningarna) det har anslutits indikatorer från ett statusord (PISW eller DSW). Se fig. 1.

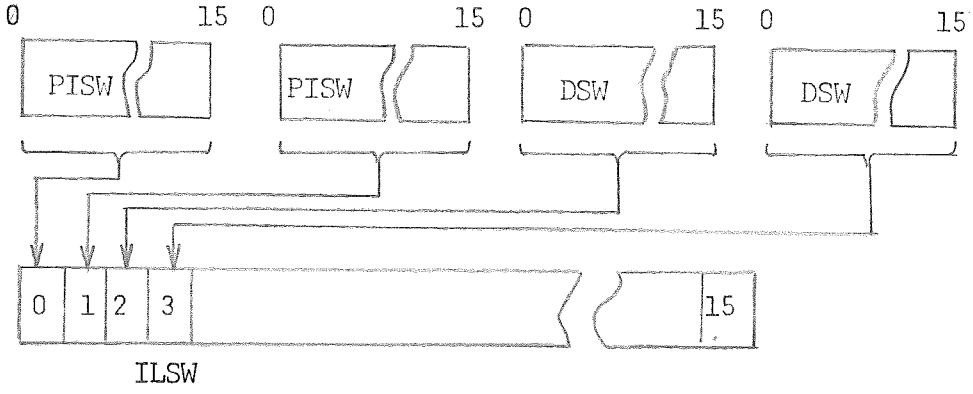


Fig. 1

Då en avbrottsbegäran detekterats inhiberar P-C den normala accessen till minnet och genererar i Storage Buffer Register en indirekt BSI (Branch and Store Instruction Register) adressinstruktion. Detta utföres med maskinvara. Adressen vid ovanstående hopp är unik för varje avbrottsnivå. Programfunktionen från denna nivå visas i fig. 2 och beskrives nedan.

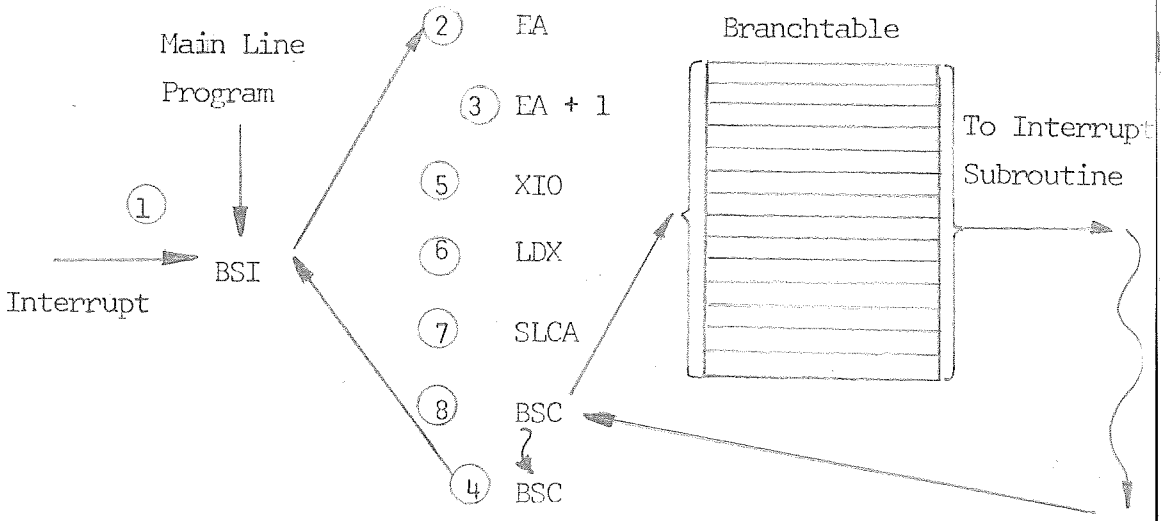


Fig. 2

1) Avbrottsbegäran inträffar under utförande av huvudprogrammet
2) Instruktionen BSI Indirect lagrar innehållet i I-registret i instruktionens effektiva adress (EA). Instruktionen medför ett hopp till avbrotts-subrutinens adress (EA + 1).
3) Avbrottssubrutinen lagrar alla data och/eller indexregister som den kommer att använda och återlagrar samma data och/eller indexregister innan subrutinen är slutförd.
4) Den sista instruktionen i subrutinen är en Branch or Skip on Condition (BOSC) instruktion som återför programmet till adressen som tidigare lagrats i EA. Denna adress är nästa instruktions läge i huvudprogrammet. BOSC instruktionen nollställer även avbrottsnivån så att nivåer med lägre prioritet kan detekteras.

Anmärkning

Eftersom ett antal avbrottsanmodanden kan tilldelas godtycklig prioritetsnivå är det nödvändigt att göra en programanalys av den anmodade prioritetsnivåns ILSW för att kunna fastställa signalens ursprung. Denna analys verkställes inom subrutiner på följande sätt (se fig. 2):

5) En instruktion XIO Sense Interrupt Level ombesörjer en inläsning i A-registret av det ILSW som för tillfället betjänas.

6) En instruktion Load Index Register (LDX) laddar index-registret med det antal avbrottssignaler som tilldelats ILSW.

7) En instruktion Shift Left and Count (SLCA) utföres. Den i registret utförda räkningen svarar mot den första 1-biten i A-registret.

8) En instruktion Branch or Skip on Condition (BSC) utföres. Denna instruktion är både indirekt och indexerad med innehållet i indexregistret. BSC-instruktionens adress är relaterad till nedersta positionen i Branch Table. Se fig. 3.

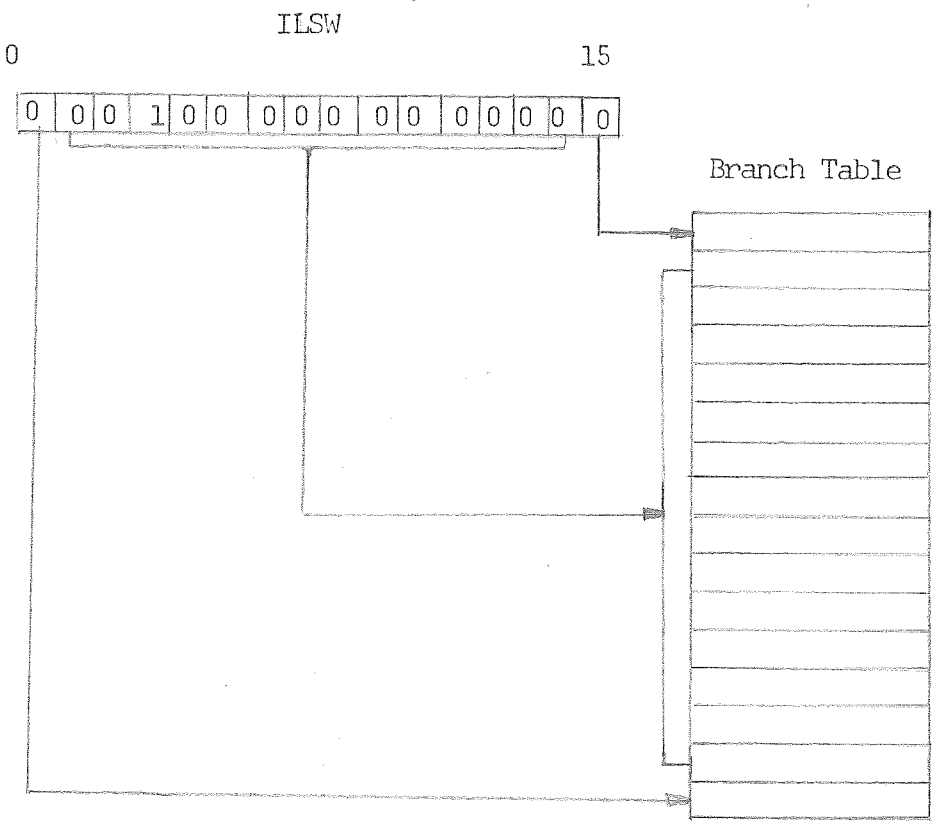


Fig. 3

Branch Table är en tabell med adresser. Varje adress leder till en avbrottssubrutin, vilken är relaterad till en avbrottsanmodan i ILSW. Om bitposition noll innehåller en etta (i ILSW) användes således sista ordet i tabellen och BSC-hoppet sker alltså till den adress som lagrats som sista ord i tabellen.

Ovanstående sekvens av instruktioner lokaliserar avbrottssubrutinen för den bit i ILSW som initierade avbrottet. Om organet som begärde betjäning (avbrott) tilldelats DSW eller PISW är det nödvändigt att bestämma vilken indikator i DSW eller PISW som är ansvarig för begäran om avbrottet. Denna identifikation kan göras nästan identisk med de tidigare beskrivna programstegen 1 t.o.m. 8 i fig, 2. Se nedan.

1) En instruktion XIO Sense Device utföres i steg 5 i stället för XIO Sense Interrupt Level. Area och/eller Modifier måste specificera organet eller statusordet.
2) Instruktionen LDX (steg 6) laddar indexregistret med maximala antalet indikatorer som tilldelas DSW eller PISW i stället för antalet avbrottsanmodanden som tilldelats ILSW.

PDP-8 /Ref. nr 18/

PDP-8 har ett tämligen enkelt brytsignalsystem med en enda nivå som avsökes med maskinvaru-utrustning. Organ som kräver omedelbar betjäning från maskinens program kan använda sig av detta system (Program Interrupt = PI). I detta status kan maskinen initiera operationer på I/O-utrustning och sedan fortsätta huvudprogrammet tills organet begär betjäning. PI-möjligheten befriar huvudprogrammet från nödvändigheten av upprepade "flag"-checkar av I/O-organen genom att låta dessa automatiskt orsaka programavbrott. När programavbrott inträffar överföres programkontrollen till en subrutin som bestämmer vilket organ som begärde avbrottet och därefter initieras en lämplig betjäningsrutin. Om endast ett organ är förbundet med PI-faciliteten kan programkontrollen vid avbrottsanmodan överföras direkt till en rutin som betjänar organet. Möjlighet finns också att via multiplexer (Type DM01 Data Multiplexer) ansluta upp till sju stycken I/O-organ.

PDP-9 /Ref. nr 19/

I/O-organen till PDP-9 kan förbindas antingen med den dubbelriktade I/O-bussen eller med Direct Memory Access (DMA) kanalen. I/O-bussen kan användas vid alla programmerade överföringar och datakanalöverföringar medan DMA-kanalen

användes för extremt snabba organ, som måste ha omedelbar access till minnet. Utgående från detta faktum har I/O-överföring av data givits följande prioritetsstruktur:

1) Begäran om DMA-kanal (högsta prioriteten)
2) Begäran om DC-kanal
3) Prioritetsavbrott (8 nivåer) - optional
4) Programavbrott
5) Huvudprogram

En anmodan om avbrott från en högre prioritetsnivå avbryter vid slutet av innevarande instruktion den pågående betjäningen eller, i fallet med DMA-kanalen, vid slutet av den pågående minnescykeln. Programavbrott och prioritetsavbrott kräver att huvudprogrammet överför kontrollen till subrutiner. Dessa subrutiner återför programkontrollen efter avslutat betjäningsintervall till det status som rådde före avbrottet. Begäran om datakanal effektueras av maskinvaran genom dataavbrott (data breaks) , d.v.s. programarbetet fördröjes medan DC-kanalen överför information mellan minnet och det aktuella organet via Memory Buffer Reg.

Låt oss litet närmare undersöka de ovan nämnda funktionerna. Vi börjar med DMA-kanalen. Denna kanal passerar ej genom CP (Central Processor) utan bildar en direkt väg mellan kärnminnet och high-speed-utrustningen. Upp till tre organ kan samtidigt betjänas med hjälp av en multiplexer, vilken upprättar prioritetsförhållanden mellan de anslutna organen. DMA-kanalen arbetar enl. "cycle stealing" principen, d.v.s. en begäran om betjäning fördröjer huvudprogrammets arbete med en minnescykel, under vilken tid ett ord överförs till eller från minnet. Programarbetet återupptages efter fullgjord överföringscykel. Slutligen bör påpekas att DMA-kanalerna har access till minnet genom var sin ingång till varje minnesmodul (8 K). Se fig.i bilaga 5.

Data-kanal (DC)-operationen använder I/O-bussen för data-överföring mellan kärnminne och yttre organ, såsom DEC

tape, magnetic tape etc. DC-systemet kan samtidigt betjäna upp till fyra organ och utökning till åtta organ är möjligt. Prioriteten upprättas med hjälp av maskinvara.

Programkontrollerad överföring utföres med hjälp av I/O Transfer (IOT) -instruktioner, som ingår i huvudprogrammet eller i betjäningssubrutiner. Dessa instruktioner är mikrokodade för att endast påverka ett speciellt organ. Mikrokoden innehåller alltså en unik organutväljande kod förutom den egentliga instruktionen.

Programavbrottsmöjligheten erbjuder en effektiv metod för betjäning av I/O. Maskinen fortsätter sitt programarbete tills ett yttre organ signalerar att betjäning kräves. I detta läge avbrytes det pågående programmet och kontrollen överförs till en betjäningssubrutin.

Som extra utrustning erbjudes ett 32-kanalers Automatic Priority Interrupt (API)-system, vilket ombesörjer prioritetsbetjäning av yttre organ. API:n har åtta olika prioritetsnivåer. Var och en av dessa äger företräde framför lägre API-nivåer, programavbrott och huvudprogram. Denna prioritetsstruktur tillåter höga datahastigheter eller kritiska organ att avbryta betjäningen av långsammare eller mindre kritiska organ med ett minimum av program overhead. De fyra högsta nivåerna är reserverade för snabb access till I/O-betjäningssubrutiner som svar på organinitierade betjäningsanmodanden. Var och en av dessa nivåer tillåter multiplexing. I detta fall innebär det att upp till åtta organ kan tilldelas samma prioritet. De fyra lägsta API-nivåerna är tilldelade processorn. I detta fall användes de till att på programinitierade anmodanden överföra kontroller till program eller subrutiner på prioritetsbasis.

S-2 /Ref. nr 20/ Nedan följer en allmän beskrivning av brytsignalsystemet för S-2. Detta erbjuder upp till 132 distinkta avbrottsnivåer,

var och en med sin egen prioritet och sin egen unika plats i minnet. När ett avbrott inträffar, identifieras dess källa och dess prioritet i relation till andra samtidiga avbrott bestämmes snabbt och automatiskt med maskinvara. Eftersom dessa funktioner inte behöver programmeras, upptar avbrottsrutiner ett minimalt minnesutrymme och funktionerna utföres även snabbt.

Varje avbrottsnivå kan individuellt bli disarmed för att sluta svara och/eller bli disabled för att uppskjuta svaret. (Ovanstående begrepp kommer att utförligare behandlas längre fram.) Denna förmåga tillåter dynamisk tilldelning av prioritetsnivåer även då man utför real-time program.

Program måste ofta skrivas för speciell utrustning som använder sig av avbrott, innen denna utrustning finns på plats. S-2 kan initiera en godtycklig avbrottsnivå med en enkel instruktion. Denna egenskap tillåter alltså små rutiner att simulera den speciella (icke tillgängliga) utrustningen vid felsökning av skrivna program.

Initiering av avbrott är också värdefullt då man upprättar en hierarki av avbrottssvar. En rutin med hög prioritet kan slutföra den brådskande delen av behandlingen och därefter trigga en lägre prioritetsnivå att uppskjuta det mindre brådskande avsnittet. Denna teknik tillåter snabbare betjäning av mellankommande prioritetsavbrott.

Vi övergår nu till att mera detaljerat undersöka avbrottssystemet. Man skiljer mellan interna och externa avbrottsnivåer. De interna avbrottsnivåerna inkluderar de som normalt levereras med S-2 systemet plus de nivåer (optional) som är knutna till räknare (counter, real-time clock), power fail-safe, memory parity, protection violation och counter-equals-zero. De interna avbrottsnivåerna är arrangerade i tre grupper:

1) counter group
2) override group
3) I/O group

Begreppen förklaras nedan.

1) Counter-gruppen består av fyra avbrottsnivåer, vilka erhåller pulser från interna och externa klockkällor. När en klockpuls mottagits av någon av counter-avbrottsnivåerna (och nivån är armed och enabled, förkl. se sid.30 och 31, i det följande) ökas värdet med 1 i den minnesposition som är associerad med den aktuella nivån och denna blir cleared och armed. Om värdet i den nämnda minnespositionen blir noll efter ökningen triggas den motsvarande counter-equals-zero avbrottsnivån i I/O-gruppen. Alla andra avbrottsnivåer behandlas av avbrottsbetjäningsrutiner och benämnes "normala" avbrottsnivåer. Counter-avbrottsnivåerna kan överföras till tillstånden armed, disarmed, enabled, disabled eller triggas vid spec. konfiguration av WRITE DIRECT instruktioner. (Se sid. 21, S-2 Computer Ref. Manual) Märk att dessa nivåer ej kan inhiberas. Prioriteten för counter-avbrottsnivån ligger under prioriteten för power off avbrottsnivån, men över prioriteten för avbrottsnivån associerad med paritetsfel vid överföringar till och från minnet.

2) I override-gruppen är alla avbrottsnivåer associerade med oberoende, valfria egenskaper. Avbrottsnivåerna i denna grupp är alltid armed (kan inte bli disarmed), enabled (kan inte bli disabled) och kan inte triggas av WRITE DIRECT instruktion. De kan inte heller inhiberas. I gruppen ingår Power Fail Safe, Memory Parity Error, Protection Violation och Multiply/Divide Exception.

   Två avbrottsnivåer betjänar Power-fail-safe och dessa användes till att hoppa in i rutiner som bevarar och återlagrar rörlig information i händelse av kraftavbrott. Power-off-nivån triggas när spänningen sjunker under en viss gräns och Power-on nivån triggas när spänningen stigit över en viss gräns.

Avbrottsnivån som betjänar Memory Parity användes till att informera programmet (eller operatören) att ett paritetsfel inträffat vid access till minnet. Om PARITY ERROR omkopplarna på panelen står i positionerna INTERRUPT/NORMAL när paritetsfelet inträffar, triggas Memory Parity-avbrottsnivån.

Avbrottsnivån som betjänar Protection Violation triggas då en kränkning av skyddat område detekterats (förutsatt att PROTECT-omkopplarens position är ON).

3) Vi övergår nu till I/O-gruppen. Denna grupp innehåller två avbrottsnivåer som standard och åtta tillvalsnivåer. Avbrottsnivåerna associerade med I/O och kontrollpanelen är standard medan counter-equals-zero och de fyra första externa avbrottsnivåerna utgör tillval.

I/O-avbrottsnivån accepterar avbrott från det I/O-system som är standard. En I/O-rutin skall innehålla en ACKNOWLEDGE I/O INTERRUPT (AIO) instruktion som identifierar källan och orsakar ett I/O avbrott.

Avbrottsnivån associerad med kontrollpanelen är förbunden med INTERRUPT-omkopplare på processorns kontrollpanel. Denna avbrottsnivå kan således triggas av operatören och tillåta honom att initiera specifika rutiner.

Counter-equals-zero avbrottsnivån är associerad med fyra (optional) reelltidsklockor. För varje installerad klocka ökar CPU automatiskt en av fyra minnespositioner när klockpulserna mottages. När värdet i en räknare är lika med noll triggas motsvarande counter-equals-zero-avbrottsnivå. Räkningen fortsätter efter triggningen om inte nivån blir disarmed eller disabled tills noll nås igen.

Vi övergår nu till att behandla de externa avbrottsnivåerna. S-2 systemet kan innehålla upp till 9 grupper av externa

avbrottsnivåer med upp till fyra nivåer i första gruppen och upp till sexton nivåer i varje efterföljande grupp. De första fyra nivåerna kontrolleras av interna avbrottsnivåer (eftersom de utgör en del av I/O-gruppen) och de har även lägre prioritet än de andra nivåerna i I/O-gruppen. Alla övriga externa avbrott kontrolleras separat och kan arrangeras i nästan godtycklig prioritetsordning.

En avbrottsnivå karakteriseras med hjälp av tre flip-flops (F-F). Två av dessa användes till att definiera ett av följande disjunkta tillstånd: disarmed, armed, waiting och active. Den tredje F-F användes till att enable eller disable (se förklaring längre fram) nivån. De olika tillstånden och omständigheterna som orsakar ändring i tillståndet beskrives här nedan.

Disarmed - När en avbrottsnivå befinner sig i detta tillstånd tillåtes ingen signal nå denna nivå, d.v.s. inget minne av signalen uppstår.

Armed - En avbrottsnivå i detta tillstånd är kapabel att acceptera och komma ihåg en avbrottssignal. Mottagandet av en sådan signal överför tillståndet för nivån till waiting,

Waiting - Nivån förblir i detta tillstånd tills den tillåtes avancera till active state. Följande villkor måste samtidigt vara uppfyllda för att nivån skall ha möjlighet att avancera:

1) Nivån är enabled (tredje F-F = on)
2) Gruppinhiberingen är off
3) Ingen högre prioritetsnivå är i enabled, active state
4) CPU befinner sig i en fas som går att bryta

Active - När en normal avbrottsnivå uppfyller alla nödvändiga krav för att tillåta ändring från waiting state till active state undanlagrar maskinen det innevarande PSD (program status double word, d.v.s. relevant information om tillståndet som rådde vid avbrottsanmodandet) i en position

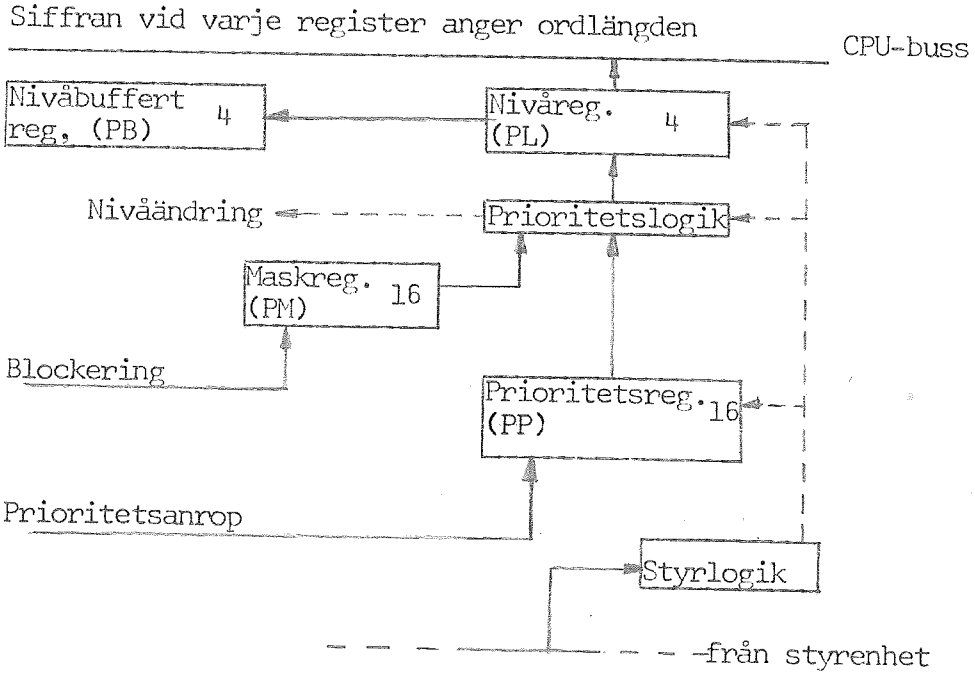
som specificerats av innehållet i det område som associerats med avbrottsnivån. Den första instruktionen i betjäningsrutinen tages sedan från den position som följer efter det lagrade PSD.

Level Enable - Den tredje F-F kontrollerar här om nivån kan avancera från waiting state till active state. Om denna F-F är off kan nivån undergå alla tillståndsändringar förutom den från waiting till active state.

UAC 1601
/Ref. nr 21/

För UAC 1601 gäller följande: Avbrottssystemet består av 16 prioritetsnivåer, där avsökningen sker med maskinvaruutrustning. För varje prioritetsnivå finns för programmeraren 16 st allmänna register tillgängliga, som är exklusiva för denna nivå. Avsökningen av dessa s.k. sub-nivåer sker alltså med programvara och tar därför lång tid (upp till 0.5 ms). De arbetande enheterna är prioritetsenheten (PU) och brytsignalenheten (ISU). PU skall med hjälp av utifrån kommande signaler bestämma den högsta prioritetsnivån på vilken maskinen kan arbeta och vid behov ge signal om nivåändring till styrenheten. ISU skall ombesörja registrering av inkommande externa brytsignaler såväl från olika process-in/ut-enheter som från processen.

PU har följande utseende:



Ettor i PP markerar anrop av motsvarande prioritetsnivå (bitpos. 0 har högst prioritet) medan nollor i PM markerar att motsvarande prioritetsnivå tills vidare skall spärras av PU. Prioritetslogiken kodar av den mest signifikanta ospärrade positionen av ettor i PP som ett 4-bitars binärtal. När detta tal ej överensstämmer med innehållet i PL ges begäran om nivåändring till styrenheten. Om ingen ospärrad etta finns ges signalen No Priority Request tills en etta uppträder.

Buffertreg. (PB) laddas med numret på den nivå maskinen lämnar vid detekterade inre fel , som orsakar anrop av systemnivån (högsta nivån).

Sådana inre fel är:

a) paritetsfel vid överföring
b) otillåten operationskod
c) spill vid multiplikation
d) fel vid division

ISU innehåller ett antal brytsignalregister (ISR), som via grindar och adressavkodare är anslutna till TD (T = transfer, D = data; databussar) resp. TA (T = transfer, A = adress; adressbussar) så att registrens innehåll kan hämtas, ett- och nollställas.

Ett ISR kan anslutas till varje prioritetsnivå. Varje ISR består av max. 16 bitar, men kan vid behov kortas av till 10 eller 4 bitar, varvid bitar i registrets minst signifikanta del avlägsnas. Bitpositionerna i varje ISR är ellerförbundna så att motsvarande bit hos PP ettställs och hålles ettställd.

2 Minnesskydd

Vi övergår nu till att betrakta olika former av minnesskydd (programskydd). Man kan här urskilja två typer. Den första typen består helt enkelt av platser i minnet från vilka endast information kan läsas (read-only) beroende på om en skyddsbit har satts eller ej. Den andra me-

toden medger off-line körning under pågående on-line-drift. Den senare metoden, som är den mera raffinerade, kommer att presenteras närmare längre fram i samband med CDC 1700.

Nackdelen med den första metoden är att den ej hindrar ett oskyddat eller off-line program att hoppa eller sekvensvis stega sig in i ett skyddat program och där utföra instruktionen som tillfälligt modifierar eller t.o.m. förstör det skyddade området. Låt oss nu undersöka hur detta problem har lösts på de utvalda maskinerna.

CDC 1700 /Ref. nr 15/

CDC 1700 har programskydd av andra typen (se ovan). Detta förefaller vara ett ganska avancerat och tillförlitligt system. Det är uppbyggt och fungerar enligt nedanstående beskrivning.

Programskyddssystemet är konstruerat kring en programskyddsbit (bit 17) som ingår i varje minnesord. Denna bit är set eller cleared av programskyddsinstruktionerna SPB (Set Program Protect) eller CPB (Clear Program Protect). Alla operand- och instruktionspositioner i det skyddade programmet skall ha denna bit set.

Programskyddet bemyndigas manuellt genom en tvåläges omkopplare på maskinens konsol. Då omkopplaren inte bemyndigar programskydd kan inga försök till kränkningar av programskyddet detekteras.

När en kränkning har konstaterats, utföres ett internt avbrott. En kränkning innebär att ett oskyddat program har försökt utföra en operation som kunnat skada det skyddade programmet. De fyra typerna av programkränkningar är:

1) Ett försök har gjorts av en oskyddad instruktion att skriva i en minnescell som innehåller en skyddad instruktion/operand. Innehållet förblir dock oförändrat.
2) Ett försök har gjorts att skriva i en skyddad minnesposition via en extern minnesaccess där en oskyddad instruktion var den ursprungliga källan till försöket. Innehållet oförändrat.

3) Ett försök har gjorts att utföra en skyddad instruktion direkt efter en oskyddad.
4) Ett försök har gjorts att utföra priviligierade instruktioner när de är oskyddade.

All perifer utrustning som är väsentlig vid användning av det skyddade programmet har en PROGRAM PROTECT-omkopplare. Om denna är tillslagen svarar motsvarande organ med att tillbakavisa alla oskyddade order. Organet svarar på alla skyddade order på normalt sätt. Då omkopplaren är i läge från svarar organet på både skyddade och oskyddade order på vanligt sätt.

Sammanfattningsvis fungerar systemet på följande sätt:

Om en skyddsbit är satt kan i motsvarande cell endast lagras information med hjälp av en skyddad instruktion. Däremot kan innehållet i cellen lagras antingen i en annan skyddad eller oskyddad cell. Om emellertid skyddsbiten inte är satt kan instruktionen i den cellen endast lagra på oskyddade platser. En skyddad instruktion kan ej följa efter en oskyddad. (Undantag från detta sker när ett avbrott inträffar och uthopp sker från en oskyddad till en skyddad subrutin.) Detta skyddar mot hopp eller sekvensvis stegning in i en skyddad area från en oskyddad rutin. Oskyddade instruktioner kan ej påverka avbrottsystemet eftersom detta är under kontroll av det skyddade programmet.

GE/PAC 4000 /Ref. nr 16/

På GE/PAC 4000 finns en form av minnesskydd, vars uppgift är att möjliggöra felsökning on-line utan att skada de arbetande programmen. Detta är en maskinvaru-egenskap som kan fås till systemen GE/PAC 4050 och 4060. Minnesskyddet lämnar möjlighet att ge restriktioner beträffande accessen till bestämda minnesareor. Detta gäller dock ej vid direktkontroll från Monitor-programmet.

Det finns tre klasser av skydd, som oberoende av varandra kan bemyndigas eller upphävas. Dessa är:

Store Protect
Branch Protect
Input/Output Protect

Då skydd har bemyndigats medför Store Protect mode att den aritmetiska enheten jämför den effektiva adressen (vid minnesrefererande adresser) med adresser för en skyddad region, vilka lagrats i Memory Fence Registers. Om operandens adress är otillåten (tillhör en skyddad area) hindras minnescykeln och överföring av programkontrollen till plats 20_8 utföres. Denna position (20_8) utgör början på monitorprogrammet. Monitorprogrammet undersöker instruktion som infångats för att besluta om den kan utföras utan risk. Om så är fallet kommer detta att ske under kontroll av monitorprogrammet; om inte, kommer en indikation att tryckas ut.

I de fall Branch Protect mode (I/O Protect mode) har anmodats orsakar detta att alla instruktioner av hopptyp (I/O-typ) inhiberas och programkontrollen överförs till 20_8. Den fortsatta sekvensen är analog med den för Store Protect ovan beskrivna.

Anledningen till transferering av kontrollen till 20_8 är att tillåta det program, som felsökts, att använda egenskaperna hos monitorprogrammet och andra program så länge detta kan göras utan att riskera fungerande program.

När minnesskyddet har upphävts finns inga restriktioner på instruktionerna Store, Branch eller Input/Output.

IBM 1800
/Ref. nr 17/

Beträffande IBM 1800 utgör dess minnesskydd endast skydd för speciella platser i minnet från felaktig lagring av information under utförandet av ett program. Detta skydd erhålles genom att förse varje position i minnet med en minnesskyddsbit. Tillståndet för varje position karakteriseras av

read only eller read/write med hänsyn till minnesskyddsbiten. Read only indikeras med 1. Read only möjliggör access till skyddad area, inläsning av innehållet i B-registret (Storage Buffer Register) och återinläsning i minnet av innehållet som lästs ut. Vilken del i minnet som helst kan under programkontroll tilldelas read only. Eftersom varje position har sin egen skyddsbit kan den således individuellt bestämmas med hjälp av instruktionen Store Status. Beträffande Store Status hänvisas till instruktionslistan. Denna instruktion användes till Write eller Clear av skyddsbiten. Utförandet av denna instruktion står under kontroll av Write Storage Protect Bits - omkopplaren på konsolen. Med omkopplaren i läge till kan instruktionen Store Status ändra skyddsbiten. När omkopplaren är i läge från utföres instruktionen som NO - OP (No-Operation).

Varje försök av programmet att skriva i read only positioner resulterar i en brytsignal på den högsta nivån.

Om en XIO eller cycle steal operation försöker att skriva i en skyddad area förblir det skyddade innehållet intakt och Storage Protect Violation - indikatorn sättes i enlighet med det DSW (Device Status Word, se sid.21) som är associerat med det organ som arbetar på datakanalen. Något internt nivåavbrott inträffar ej.

Check Stop omkopplaren (på konsolen) i läge ON förorsakar att Process-Controlern (P-C) stannar vid slutet av en minnescykel i vilken en Storage Protect Violation (SPV) har detekterats. Då omkopplaren står i läge OFF förorsakar SPV P-C att initiera ett internt avbrott eller en laddning av SPV-indikatorn med tillhörande DSW som beskrivits ovan.

Omkopplaren Disable Interrupt på konsolen i läge ON förhindrar ett internt avbrott.

PDP 8 Minnesskydd saknas

PDP 9 Minnesskydd saknas

S-2 /Ref. nr 20/

Det primära ändamålet med skyddssystemet (Protection System) är att garantera integriteten mellan ett förgrundsprogram och ett bakgrundsprogram då dessa utföres samtidigt. S-2 systemet erbjuder både operationsskydd (se längre fram) och skydd mot skrivning i skyddat område med hjälp av ett 16-ords register (optional). Varje bit i dessa 16 ord är associerad med ett specifikt block i kärnminnet. Ett block i kärnminnet består av en region av 256 konsekutiva celler, vars lägst numrerade adress är en heltalsmultippel av 256. Således är bit 0 i skyddsregister 0-associerad med minnesplatserna 0 till X´FF´ (uttryckt i hexadecimal form med X som utfyllnad) och bit 1 i skyddsregister 0 associerad med minnesplatserna X´100´ till X´1FF´ och bit 15 i skyddsregister X´F´ är associerad med minnesplatserna X´FF00´ till X´FFFF´. Värdet 0 av en bit i skyddsregistret betyder oskyddat minnesblock och värdet 1 betyder följaktligen skyddat block.

Skyddsregistren kan laddas individuellt genom WRITE DIRECT instruktionen (se sid. 21, S-2 Computer Ref.Manual) med en effektiv adress av formen X´8 r´ där r är en hexadecimal siffra som bestämmer vilket skyddsregister som skall laddas från ackumulatorn. Således kan skyddsbitarna för 16 minnessidor (memory pages), 4096 ord, sättas upp med hjälp av en enkel instruktion.

Körning med skyddssystemet står under kontroll av PROTECT-omkopplaren och nyckellåset på processorns kontrollpanel. Om skyddssystemet är i funktion gäller följande regler:

1) Priviligierade instruktioner kan utföras endast om de erhållit access från skyddat minne. Om en priviligierad instruktion erhåller access från oskyddat minnesutrymme utföres inte instruktionen, utan i stället triggas avbrottsnivån associerad med protect violation.
2) En instruktion hämtad från ett oskyddat minnesutrymme kan omedelbart följas av en instruktion hämtad från ett skyddat minnesutrymme endast då det gäller som ett svar på en avbrottsanmodan.

3) Instruktionerna STORE ACCUMULATOR (STA) eller INCREMENT MEMORY (IM) kan användas till att ändra skyddat minne bara om instruktionen får access från skyddat minne. Om ett försök göres att ändra skyddat minne med en instruktion som fått access från oskyddat minne utföres ej operationen. I stället triggas avbrottsnivån associerad med protect violation.

UAC 1601 — På UAC 1601 finns f.n. ingen form av program- eller minnesskydd. Problemen penetreras och framtida versioner kommer med all sannolikhet att förses med någon form av minnesskydd.

3 A/D- och D/A-omvandlare

Då man skall integrera ett DBS i en konventionellt reglerad processindustri går det ofta att behålla de gamla givarna, kontrollinstrumenten och ställdonen. Man får dock göra en analys av dessa beträffande tillförlitlighet, underhåll, möjligheter till att bygga in kontrollfunktioner, manuell parallellkörning och kostnader för ev. byte till ny, bättre utrustning.

Vid inpassningen av ett digitalt DMS i en process måste man omvandla de analoga signalerna från processens givare och mätinstrument till digitala signaler, passande för CPU. Denna konvertering göres i A/D (analog/digital) omvandlare. Motsvarande omvandling från CPU´s digitala utsignaler till analoga signaler sker i D/A (digital/analog) omvandlare, se fig. 1.

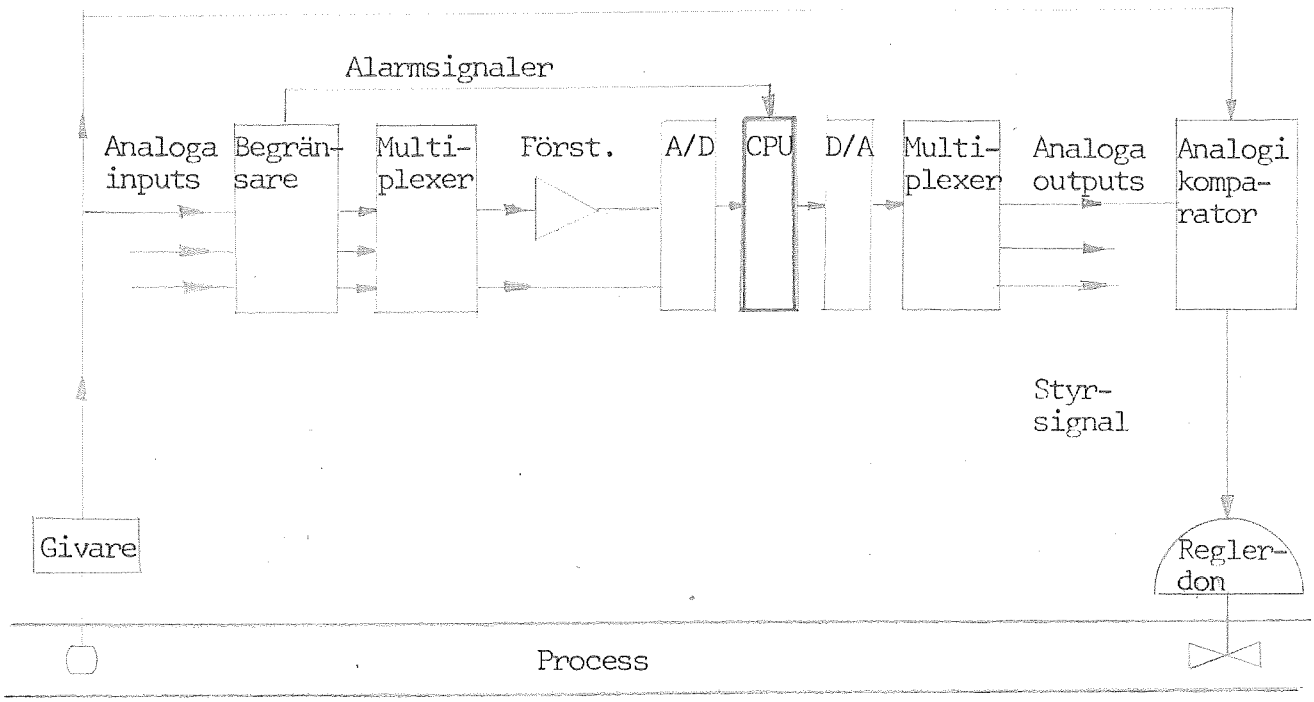


Fig. 1

Signalkretsar i PDMS /Ref. nr 3/

De analoga process-signalerna filtreras (t.ex. genom max/minbegränsare eller -varnare), multiplexas, förstärkes, om det är nödvändigt, och omvandlas till digital form. Vid övervakande system används de omvandlade utsignalerna som setpoint-signaler i analogikomparatorer. Om man har DDC, går de analoga signalerna direkt från multiplexern (om sådan finns) ut till reglerdonen.

A/D-omvandlare

De flesta A/D-omvandlare har en elektrisk spänning som insignal. Denna signal kan genereras av olika typer av givare, t.ex. termoomvandlare, pneumatisk givare med lämpliga omvandlare, rena elektriska givare, strömningsgivare med omvandlare.

Det finns många olika tekniska principer för A/D-omvandling. En grupp omvandlare överför den inkommande signalen till en förskjutning i rummet som i sin tur direkt överförs på digital form, se fig. 2.

kontaktborstar svarande
mot de 4 bitpositionerna

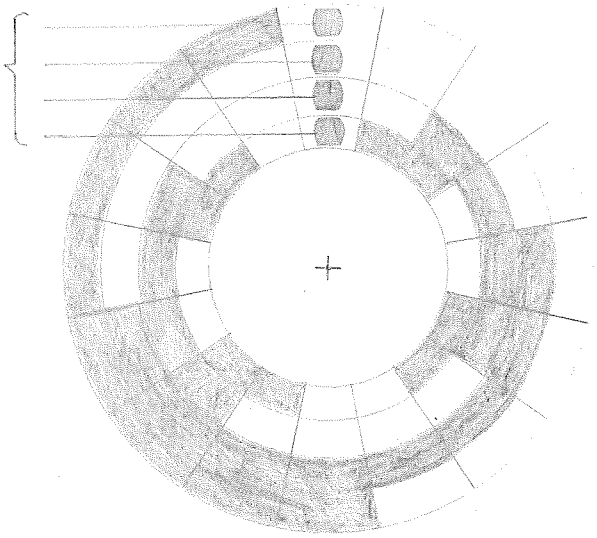


Fig. 2

Exempel på A/D-omvandling /Ref. nr 7/
Här har en insignal via ett vridspoleinstrument överförts till en vridning av den binärkodade skivan som direkt via kontaktborstarna överför vridningsvinkeln till en binärkod

En annan grupp använder komparatormetoder, se fig. 3.

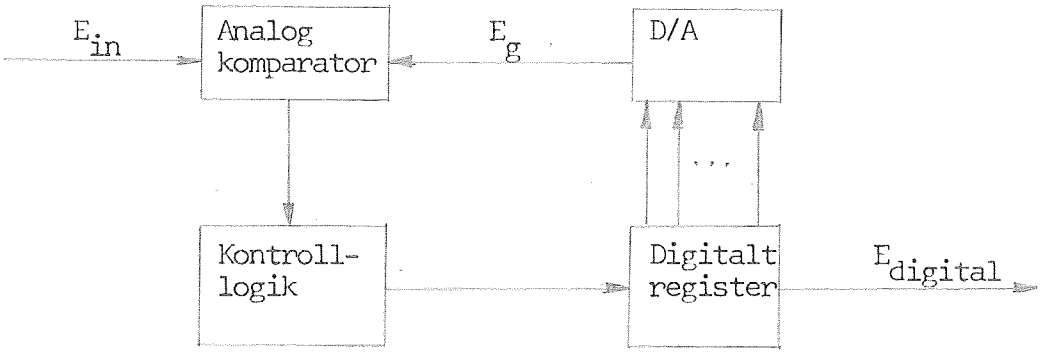


Fig. 3

Exempel på en successiv approximationsmetod för A/D-omvandling
Här jämförs insignalen med en internt genererad spänning (E_g). När de två signalerna är lika, läses det digitala registret. Här fordras en D/A-omvandlare i feedbackloopen.

De olika givarna i processen känns av enligt ett visst program. Avläsningsfrekvensen kan variera från 10000 tal/sek till en avläsning/dag, beroende på hur snabbt processförloppet är och den önskade noggrannheten.

En grupp omvandlare som på detta sätt är lämplig för stickprovsmätningar är spännings/frekvens- och spännings/tid omvandlare, se fig. 4 resp. fig. 5. /Ref. nr 27/

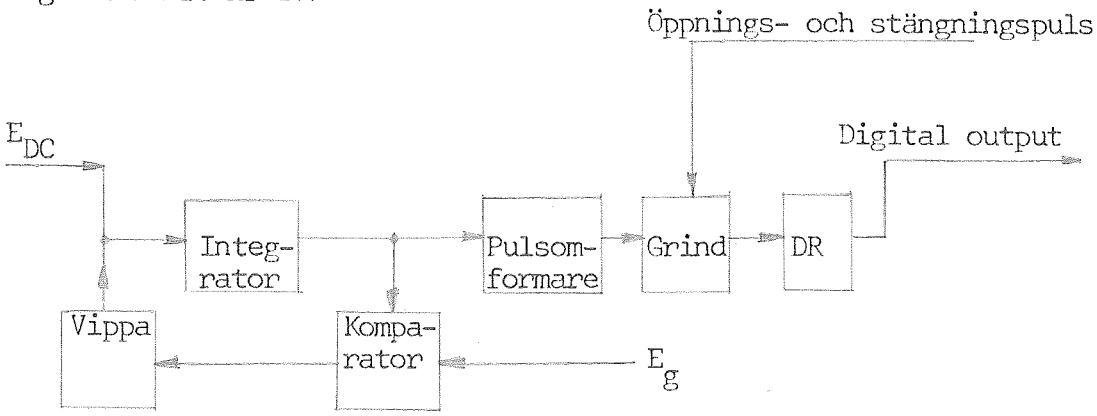


Fig. 4

Spännings/frekvens omvandlare

Den analoga DC-signalen (E_{DC}) integreras och jämförs med en referensspänning (E_g) i komparatorn. Då den integrerade rampsignalen är lika stor som referensspänningen kantrar vippan så att integratorn nollställs och samma procedur upprepas igen. Detta medför att vi från integratorn får en sågtandsspänning, vars frekvens är proportionell mot insignalens storlek. Sågtandssignalen omformas till ett pulståg som via grinden under en viss öppningstid räknas i DR, där vi nu har en digital signal.

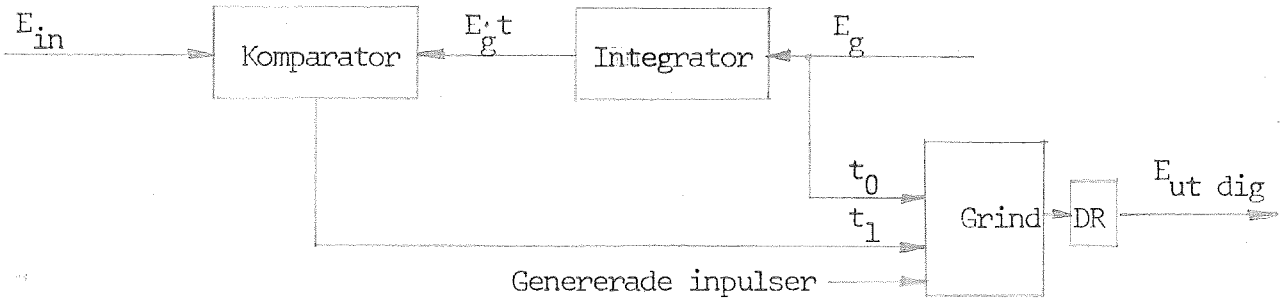


Fig. 5

Spännings/tid omvandlare

Den genererade rampsignalen ($E_g t$) jämförs i komparatorn (som kan vara en nivådetektor av differentialförstärkartyp) med den analoga DC-insignalen (E_{in}). Komparatorn styr grinden, som under jämförelsetiden släpper igenom de genererade räknepulserna till räknaren, där vi nu direkt kan ta ut digitala signaler.

För att hålla nere kostnaderna för A/D-omvandlare i ett system brukar man ha multiplexa ingångar till dessa.

D/A-omvandlare

Den till A/D-omvandling omvända proceduren, D/A-omvandling,är ofta mycket enklare att genomföra. Billigare kretsar medför här att man slopar time sharing (multiplexing) och i stället används oftast en D/A omvandlare till varje utgång.

Bland de många typer av D/A omvandlare som finns vill vi visa den vanligaste, se fig. 6.

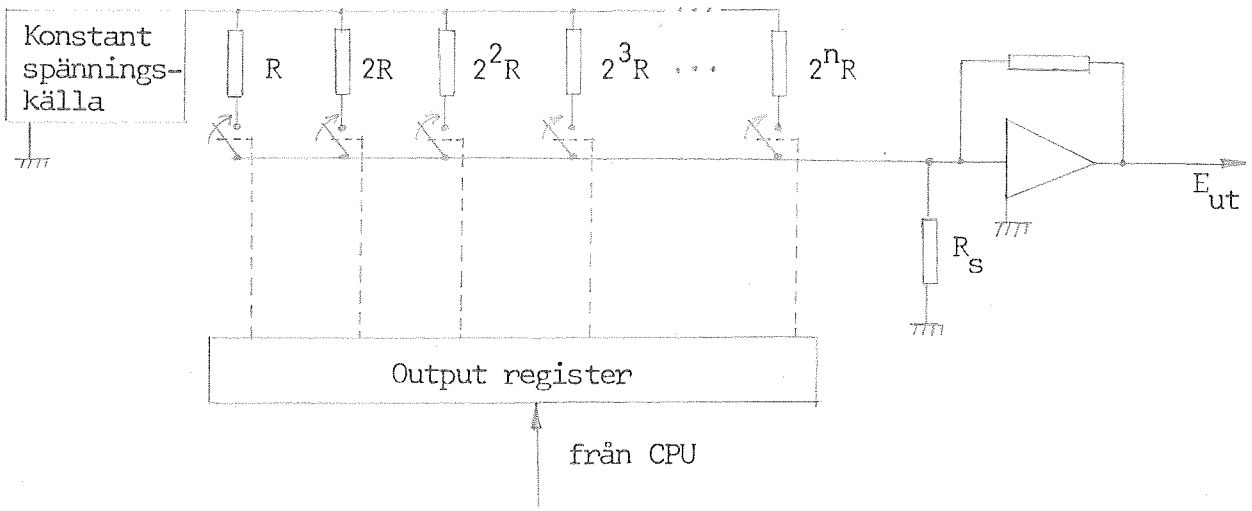


Fig. 6

Den vanligaste formen på en D/A omvandlare /Ref. nr 3/
Innehållet i outputregistret kontrollerar de snabba brytarna som öppnar eller sluter branscherna i nätet med binärviktade motstånd. Den summaström som fås genom de öppna brytarna passerar R_s och vi får alltså en mot binärsignalen i registret proportionell inspänning till operationsförstärkaren, som ger en utstpänning (E_{ut}) i analog form.

Då D/A omvandlaren ovan har en egen spänningskälla, kvarhålles utsignalen om CPU skulle falla ifrån. Här har man då möjlighet att smidigt gå över till manuell betjäning.

Sample and Hold kretsar /Ref. nr 2/

Då man skall använda A/D omvandlare vid mycket snabba förlopp med hög samplingsfrekvens, uppstår problemet att insignalen kanske varierar under själva omvandlingstiden och att man inte vet exakt tidpunkt för mätningen. För att lösa dessa problem använder man Sample and Hold enheter.

Dessa Sample and Hold kretsar består av en kondensator, ett brytsystem och förstärkare. När Sample and Hold är i läsläge, driver den analoga insignalen en förstärkare som laddar kondensatorn. Spänningen över kondensatorn följer insignalen. Då Sample and Hold kretsen får en läs- eller hold-signal kopplas kondensatorn bort från insignalen och förbinds med A/D omvandlarens ingång. En förstärkare brukar inkopplas mellan kondensatorn och A/D omvandlaren för att hindra kondensatorn att urladdas under omvandlingstiden.

Omvandlingstider för de olika fabrikanternas A/D omvandlare, se bil. nr 4.

4 Programpaket

Låt oss slutligen göra en snabböversikt av vad som erbjudes då det gäller programpaket. Det är inte alltid säkert att en detaljerad presentation görs av de olika delarna i paketet, speciellt inte då namnet tydligt anger syftet, I det fall en mera ingående beskrivning önskas hänvisas till resp. manualer.

En allmän tendens till ökande tillgång på programpaket kan skönjas. Det förefaller synnerligen gynnsamt att utnyttja denna i stället för att försöka skräddarsy sina program i alltför hög grad. Ett känt och ibland pinsamt faktum är att programarbetet underskattas med en ofrånkomlig kostnadsstegring som följd.

CDC 1700 /Ref. nr 25/

CDC 1700 innefattar i sitt programpaket följande program:

Monitor
Job Processor
Fortran Compiler
Macro Assembler
Relocatable

Breakpoint Program
Recovery Program
Library Editing Program

Monitorprogrammet tjänar som koppling mellan program och maskinvara. Det tilldelar programmen tid på prioritetsbasis.

Job Processorn kontrollerar alla off-line-program som står under kontroll av 1700-systemet.

Recovery Program är ett felsökningshjälpmedel med vilket programmeraren bestämmer kärnminnets och massminnets tillstånd vid slutet av en job execution.

Library Editing Program användes till att ändra system- och programbiblioteket.

GE/PAC 4000 /Ref. nr 16 och 22/

I GE/PAC 4000 systemet delar man upp programpaketen i tre grupper:
Program Preparation Aids
Standard On-Line Functions
Utility and Debugging Aids

Program Preparation Aids erbjuder Process Assembler Language (PAL), Fortran och Tabular Sequence Control (TASC).

GE:s FORTRAN II kompiler för GE/PAC 4000 översätter FORTRAN II till PAL. Det är tillåtet att blanda FORTRAN II uttalande med PAL-kod.
TASC-systemet består av TASC-språket, TASC:s assemblerprogram och en Table Analyzer Subroutine. Systemet är utformat för att effektivt kunna producera program för kontroll av processer av sekvenskaraktär (shut-up och shut-down)

Beträffande Standard On-Line Functions kan Monitorsystemet nämnas. Dess huvudsakliga uppgifter är tidmätning, disponering av systemaktiviteter, trumma - kärnminne - överföring och I/O-kommunikation.

Utility and Debugging Aids består av Load Routines, Dump Routines och Memory Change Programs.

Load Routines ombesörjer laddning av program och data i minnet med hjälp av hålrems- eller kortläsare.

Dump Routines skriver ut minnesinnehållet på skrivmaskin eller radskrivare eller stansar ut minnesinnehållet på hålremsa i standard loading formats.

Memory Change Programs ändrar innehållet i specificerade minnesplatser i enlighet med gjorda anmodanden. Detta kan ske med hjälp av switchar på konsolen eller I/O-skrivmaskinen.

IBM 1800 /Ref. nr 26/

IBM 1800 har följande program och programsystem att erbjuda:

Assembler
Fortran
Subroutine Library
Utility Routines
Time-Sharing Executive System

Assemblerspråket tillåter programmeraren att skriva källprogram på ett symboliskt språk som är lättare att använda än maskinspråket. Makroinstruktioner är inkluderade i assemblern. Dessa ombesörjer automatiskt hopp till subrutiner.

Subroutine Library består av rutiner som användes i samband med input/output av data, omvandling av data och aritmetiska uppgifter.

Utility Routines består av följande rutiner:

Input/Output Routine
Dump Routines
Console Routines
Loder Routines
Keyboard Routines
Disk Initialization Routine
Core Image Converter

Card Reproducing Routine
Construct Paper Tape Routine

Input/Output Routine ombesörjer överföring av data från ett medium till ett annat. Input accepteras från hålkort, hålremsa eller magnetband. Output kan ske till radskrivare, hålremsa, magnetband eller hålkort.

Dump Routines användes för att mata ut hela eller en del av innehållet i kärnminne till output-organ.

Console Routine är ett hjälpmedel vid testning av program och tillåter dumpning av utvalda delar av kärnminnet.

Keyboard Routines användes som hjälp vid utsortering och presentation av källdokumentation på hålkort eller hålremsa.

Core Image Converter användes för att konvertera assembler- och kompiler-program till kärnminnet med hjälp av Core Image Loader.

Paper Tape Construct Routine tillåter användaren att kombinera remsor (tapes), utplåna subrutiner från subrutinbiblioteket och att reproducera remsor.

Time-Sharing Executive System är ett monitorprogram för reelltids processkontroll. Med hjälp av detta system kan användaren:

1. Skriva processkontrollprogram i Fortran- eller assemblerspråk
2. Samtidigt utföra både processkontrollprogram och icke-processkontrollprogram
3. Simulera processavbrott och process-input/output för att möjliggöra testning av program utan att använda den fysiska processen eller I/O-organen.
4. Utmärka program som skall minnesskyddas med hjälp av programmering

PDP-8 /Ref. nr 18/ Som standard ingår i programsystemet följande:

MACRO-8 Symbolic Assembler
Fortran System Compiler
Symbolic On-Line Debugging Program
Symbolic Tape Editor
Floating Point Package
Mathematical Subroutines
Utility- och Maintenance program

MACRO-8 Symbolic Assembler, som accepterar källprogram skrivna i symboliskt språk, konverterar minnespositioner, maskininstruktioner och operandadresser från symboliskt språk till binär form. Den producerar även en objektprogram-tape och en tabell som definierar placeringen i minnet.

Fortran System Compiler innehåller de instruktioner som maskinen behöver för att utföra en översättning av det i Fortran formulerade problemet till ett objektprogram i maskinspråk. Kompilatorn producerar även felutskrift. Efter kompileringen laddas maskinen med objektprogrammet, styr programmet (the operating system) och de data den kommer att arbeta med för att lösa problemet. Man bör observera att objektprogrammet kan plockas ut om så önskas.

Symbolic On-Line Debugging Program medger att man med användning av Teletype keyboard/reader och teleprinter/punch kan kommunicera med PDP-8 på källspråket. Användaren kan kontrollera utförandet av varje del av sitt objektprogram genom att placera ut avbrott. När maskinen når en avbrottspunkt överföres kontrollen av objektprogrammet till Dynamic Debugging Tape (DDT). Därefter kan innehållet i individuella kärnminnesregister modifieras för att erhålla ett förbättrat objektprogram.

Symbolic Tape Editor- program användes till att skriva ut, rätta och uppdatera program, skrivna i symboliskt språk, genom att använda PDP-8 och konsolskrivmaskinen. Med detta program i kärnminnet läser användaren in delar av sin hål-

remsa (skriven symboliskt), avlägsnar, ändrar eller lägger till instruktioner eller operander och får tillbaka en ny symbolisk hålremsa med felen avlägsnade.

Floating Point Package tillåter PDP-8 att med hjälp av programvara utföra sådana aritmetiska operationer som inte kan realiseras med maskinvara.

Mathematical Subroutines kan utföra följande operationer i både enkel och dubbel precision:

addition, subtraktion, multiplikation, division, kvadratroten, sinus, cosinus, arctg, e-log och potensräkning.

PDP-8's Utility-program ombesörjer utskrift på skrivmaskin, radskrivare och hålremsa av kärnminnets innehåll i oktal, decimal eller binär form beroende på användarens önskemål. Dessutom erbjudes subrutiner för oktal eller decimal dataöverföring och konvertering binärt till decimalt och decimalt till binärt.

Beträffande Maintenance-programmet finns det en mängd standard-testprogram för att förenkla och underlätta testning och felsökning.

PDP-9 /Ref. nr 23/

Två fullständiga programvarusystem och ett underhålls- (maintenance) system finns tillgängliga för PDP-9. Programvarusystemen benämnes Basic Software Package (BSP) och Extended Programming System. Nedan följer en översikt av vad dessa innehåller och vilken maskinkonfiguration som är lämplig i de båda fallen.

PDP-9 BSP är konstruerad att arbeta med basic PDP-9 med ett kärnminne på 8 K, remsläsare, remsstansar och konsolskrivmaskiner. Man må observera att BSP är kompatibel med tillgänglig programvara för PDP-7. BSP innehåller en Fortran II Compiler, Symbolic Assembler, Dynamic Debugging

Technique (DDT) program och en Symbolic Tape Editor, alla konstruerade att arbeta såsom stand alone system (se fig. bil.5) med I/O hålremsa. Här följer en kort presentation av de olika programmen med angivande av speciella faciliteter.

Objektprogram skrivna i Fortran II kan använda extra minne för att lagra fixa fält och COMMON-fält och skriva ut på radskrivare. Huvudkomponenterna i Fortran II-systemet är:

Compiler
FORTRAN Assembler
Object-Time System (se manual)
Library, inkl. rutiner för I/O 6-dec.-digit. aritmetik och 9-dec.-digit. aritmetik
Linking Loader

Kompilatorn accepterar program skrivna i Fortran II och producerar sådana i ett övergångsspråk acceptabelt för FORTRAN assemblern. Assemblern producerar i sin tur en binär relokerbar version av programmet plus en binär version av Linking Loader.

När man är klar att köra ett program laddas maskinen med huvudprogrammet och de subprogram som behövs, följda av de standard funktioner som finns i subrutinbiblioteket. Object-Time systemet innehåller ett interpretativt system för flytande räkning, en översättare för utsagor om formaten, rutiner för t.ex. konvertering från fix till flytande räkning och I/O rutiner. Object-Time systemet upptar 4000_{10} positioner och måste finnas i minnet då ett FORTRAN II program utföres.

Bland egenskaperna hos FORTRAN II systemet kan nämnas följande:

1. Möjligheter till blandning av utsagor skrivna i assemblerspråk och Fortran
2. Konstanter i flytande räkning kan ges med antingen 6 eller 9 siffrors noggrannhet
3. Konstanter i fix räkning kan ha värden mellan ± 131071
4. FUNCTION och SUBROUTINE utsagor finnes

PDP-9's Symbolic Assembler (SA) är av one-pass typ och översätter alltså källprogrammet till binära objektprogram. Trots att assemblern är konstruerad att arbeta med minimal konfiguration (8K) kan den assemblera program för godtycklig PDP-9 konfiguration. Under assembleringsfasen löses källprogrammet rad för rad och konverteras till korrekt binärt format. Mnemoniska koder översättes till sina binära ekvivalenter och symboliska adresser ersättes med verkliga adresser. Vid slutet av assembleringen rapporteras alla påträffade odefinierade symboler. Eftersom PDP-9 SA är ett one-pass system påträffas ofta symboliska adresser innan de är definierade. Sådana symboler noteras på binary output tape och deras ekvivalenter trycks också ut på tape när de definierats. Under laddning av ett binärt symboliskt assemblerprogram svarar programmet Linking Loader för erhållandet av en tabell med odefinierade symboler och det ersätter dem med värden då de påträffas.

DDT-programmet för PDP-9 är ett mångsidigt verktyg vid kontroller och modifieringar av program. Det upptar 2000_8 celler i kärnminnet och det tillåter operatören att ladda ett program och köra hela eller utvalda delar av det under strikt kontroll. Med DDT kan avbrott införas eller hävas, register kan undersökas och ändras, patches (bitar?) kan införas och stansas ut i en form som accepteras av the loader och utskrift kan erhållas vid kritiska punkter.

Symbolic Tape Editor är konstruerad för att underlätta utarbetandet och rättandet av symboliska källprogram på magnetband. Editorn läser in delar av den symboliska tapen i minnet och gör dem på så sätt tillgängliga för undersökning, rättning och utstansning. Rättelser, ny text och order till editor programmet göres via konsolskrivmaskin. I/O-tape specificeras för att antingen ges i ASCII- eller FIODEC-kod. Information som skall listas lagras i en textbuffert i minnet. Denna buffert upptar hela det minnesutrymme, som inte användes av editorprogrammet.

Vi övergår nu till att betrakta PDP-9 Advanced Software (AS). Detta programpaket inkluderar en Fortran IV compiler, Makro Assembler, On-Line Debugging System, Symbolic Editor, Peripheral Interchange Program, Linking Loader, I/O-Programming System och Monitor. Två versioner av detta slag finns tillgängliga; ett enkelt I/O monitorsystem med remsa I/O eller kortläsare, vid användning av grundkonfigurationen och ett mera sofistikerat monitorbaserat organoberoende system för PDP-9 S med en eller flera former av yttre minnen. Båda systemen är kompatibla.

Låt oss först betrakta Paper Tape (el. Card)systemet. PDP-9 systemet med Basic (8K) eller Extended Memory utan yttre minne använder sig av Fortran IV kompilator, Makro Assembler (MACRO-9), Debugging System (DDT-9), Symbolisk Editor, Peripheral Interchange Program (PIP-9) och en Linking Loader under kontroll av I/O-monitorn. Alla system är fullständigt relokerbara och handhar eller producerar relokerbara koder.

Det organoberoende systemet kräver däremot att det finns yttre minne. Detta kan bestå av antingen

2 Dectape transports (TU 55) och kontroll (TC 02) eller
2 IBM-compatible transports (TU 20) och kontroll (TC 59)eller
1 disc system och kontroll eller
1 trum system och kontroll (RM 09)

Med tillägg av ett av dessa yttre minnen, Keyboard monitorn (KM-9) och I/O Programming System (IOPS) kan PDP-9 systemet användas för att automatiskt lagra, söka, ladda och utföra PDP-9 program. Användaren kan också inkalla sina systemprogram från godtyckligt yttre minne, kompilera eller assemblera från godtycklig input till lämplig output och (optionally) erhålla utskrift på godtyckligt skrivorgan eller på magnetband.

Komponenterna i PDP-9 Advanced Software beskrives här nedan i korthet.

Fortran IV Compiler är ett two-pass system som accepterar satser skrivna i fortran och producerar relokerbara objektkoder som laddas med hjälp av Linking Loader. Denna kompilator är fullständigt kompatibel med USA FORTRAN IV med undantag av följande egenskaper:

1. Komplex räkning ej möjlig
2. Ändring av ett fälts dimension ej möjlig
3. Enda namngivna COMMON som kan förekomma är COMMON /Data/ i övrigt används blank COMMON
4. Datasatsen kan ej innehålla implementerad DO-sats

Subrutiner skrivna i antingen Fortran IV eller Macro Assembler språk kan laddas med och inkallas av huvudprogrammen skrivna i Fortran IV.

DDT-9 (debugging system) erbjuder den flexibilitet som erhålles då Basic DDT kompletteras med relokerbarhet och reelltidsoperation. Med detta system kan man ladda och arbeta med program i reelltidsomgivning medan man upprätthåller strikt kontroll av körningen för varje sektion. Se för övrigt tidigare beskrivning av Basic DDT.

Symbolic Editor för PDP-9 med Advanced Software har förmåga att läsa symbolisk text från godtyckligt inorgan, att undersöka och rätta den och att ge utskrift på godtyckligt utorgan. Den kan även användas till att skapa nya symboliska program.

Vi övergår nu till Peripheral Interchange Program (PIP-9). Den primära funktionen för PIP-9 är att underlätta behandlingen och överföringen av datafiler från godtyckligt input- till godtyckligt outputorgan. PIP-9 kan även användas till att uppdatera filbeskrivningar, utradera, införa eller

kombinera filer, utföra konvertering av koder, återspola magnetband och slutligen återupprätta minnesutrymme som inte längre behövs av fil-organ.

Linking Loader har till uppgift att ladda ett godtyckligt Fortran IV eller Makro-9 objektprogram vilka kan föreligga antingen i relokerbar eller absolut form. Linking Loader har till uppgift att ladda maskinen med ett godtyckligt FORTRAN IV eller objektprogram (MACRO-9), vilka kan föreligga antingen i relokerbar eller absolut form. Den skall alltså ladda och relokera program, ladda inkallade subrutiner samt söka och ladda fasta subrutiner och IOPS rutiner.

IOPS (I/O Programming System) ombesörjer en standardiserad programkontakt med alla I/O-organ i PDP-9 systemet. IOPS består av en modulsamling av relokerbara I/O och utility subrutiner, vilka sänder data till och från yttre organ och gör data tillgängliga för behandling. IOPS´s data- och filbehandlingsrutiner inkluderar organoberoende för alla systemprogram. Programsystemet eliminerar också fullständigt nödvändigheten för programmeraren att programmera de yttre standardorganen. IOPS är kodad på modulbasis för att möjliggöra tillägg av nya organ och modifieringar till hanteringen av de innevarande organen. Det fordras avsevärt mindre arbete att modifiera ett IOPS subprogram än att revidera individuella I/O sektioner till att passa en ny konfiguration. Genom att använda IOPS kan man alltså snabbare ändra program så att de passar en expanderande konfiguration.

Den enkla I/O monitorn inkluderar organbestämningstabeller (device assignment tables) och I/O-rutiner som är nödvändiga för körning av program på PDP-9 S utan yttre minne. Dess funktion är att handha inputs och outputs för systemprogram och användarprogram så att programmeraren inte behöver befatta sig med organ- eller databehandlings-

rutiner. Normalt tillåtes endast hålremsa-I/O eller hålkort-input av ovannämnda monitor.

Keyboard Monitorns huvudsakliga funktion är att tillåta användaren att få direkt och omedelbar access till program lagrade i yttre minnen, underlätta upprättandet och lagrandet av nya program och att tillåta I/O-oberoende programmering genom att specificera in/ut-organet vid laddningstillfället. Monitorn innehåller även Device Assignment Tables (DAT). Ändamålet med DAT är att relatera logiska I/O enheter med det verkliga DMS. Varje input- eller outputreferens inom ett system eller användarprogram refererar till en position i en DAT. Denna tabell i sin tur innehåller en organtilldelan för varje tabellinhopp. Eftersom innehållet i tabellen kan ändras genom order till Keyboard Monitorn kan aktuella I/O organ ändras utan att programreferensen till dessa ändras.

S-2

/Ref. nr 24/

S-2's programsystem arbetar i tre olika konfigurationer:

Stand Alone (SA)
Basic Control Monitor (BCM)
Real-Time Batch Monitor (RBM)

SA och BCM köres med ett kärnminne på 8 K och utan skrivminnet Rapid Access Data (RAD) fil. BCM-konfigurationen är mera flexibel och lättare att använda än SA, men däremot tillåter SA användaren att nyttja sig av 2500 fler ord.

RBM-konfigurationen kräver ett kärnminne på 12 K samt tillgång till RAD. I denna konfiguration befinner sig monitorn, assemblern och kompilatorn på RAD och delar av dessa läses in i kärnminnet när de behövs. RAD användes också till att lagra tabeller som genererats av assemblern och kompilatorn.

De olika konfigurationerna med tillhörande programvara illustreras i bilaga nr 5.

Programsystemen är konstruerade i modulform. Detta innebär att en användare som utökar sin maskinkonfiguration kan använda större och kraftfullare delar av den existerande programvaran.

Vi övergår nu till att mera i detalj beskriva BCM och RBM. BCM underlättar multianvändning på en liten maskinkonfiguration. Stand-alone förfarande eller enkel satsbearbetning kan pågå i bakgrunden samtidigt som reellтidsbehandling äger rum i förgrunden. De allmänna BCM egenskaperna kan betraktas nedan:

1. Reellтidsprocesser är direkt sammankopplade med BCM
2. BCM utför och kontrollerar alla priviligierade funktioner inkl. I/O, avbrott och minnesskydd
3. Användarens I/O-funktioner utföres via I/O anrop till monitorn
4. Fullständig I/O mellanlagring (buffering är möjlig)
5. Fullständigt minnesskydd finns ombesörjt för operating system och förgrundsprocessen
6. BCM betjänar bakgrundsprogrammen på basis av tillgängligt minnesutrymme och behandlingstid
7. Bakgrundsuppgifter kan laddas, utföras eller avslutas när som helst utan att interferera med reellтidsoperationerna i förgrunden
8. Bakgrundsuppgifter kan köras sekvensiellt p.g.a. självinitiering hos t.ex. GE Basic FORTRAN, Symbol Assembler etc.

Om S-2´s maskinkonfiguration inkluderar RAD kan man använda sig av en egenskap som benämnes RAD Check Point. (Denna står under kontroll av BCM.) Denna egenskap möjliggör att förgrundsprocessen eller operatören kan initiera en tillfällig överföring av bakgrundsprocessen till ett skivminne. Förgrundsprocessen kan då använda hela kärnminnet till att bearbeta ytterligare förgrundsprocesser och därefter återföra bakgrundsprocessen till kärnminnet.

Vi övergår nu till att undersöka RBM litet närmare. Denna monitor förenklar det praktiska handhavandet av maskinen, och möjliggör dynamisk behandling av förgrundsprogrammet. En sammanfattning av egenskaperna följer nedan.

1. Mångsidig operatörskontroll av systemfunktioner, automatisk inmatning av job (sekvensinmatning), dynamisk kontroll av reelltidsprocesser och samtidigt utförande av bakgrundsuppgifter
2. Automatisk beräkning av arbetstid
3. Maximalt utnyttjande och handhavande av yttre minne (RAD)
4. Möjlighet till check-point betjäning
5. Användning av overlay -teknik (se nedan) för att ombesörja större och mera komplexa uppgifter
6. Möjlighet till programsimulering av ännu icke installerad maskinvara
7. Identifiering och behandling av inkommande prioriterade arbeten
8. Omfattande hjälpmedel för "konstruktion" av program, debugging och processorns arbete i form av GEC FORTRAN IV, Extended Assembler och Utility rutiner såsom felsökning och laddning
9. Fullständiga diagnostiseringsmöjligheter
10. Modulutförande för att tillåta användaren att modifiera standardmonitorn

RBM erbjuder en utvidgning av egenskaperna som innefattades i BCM. För det första erbjuder den större flexibilitet genom att göra det möjligt för förgrundsprogrammet att utvidga och återtaga sin omfattning dynamiskt med hjälp av multipla prioritetsnivåer. För det andra erbjuder den ökad effektivitet då det gäller satsvis bearbetning. För det tredje använder den sig av yttre minne (RAD) för att utöka möjligheterna till förgrundsbehandling med användning av check point egenskapen. Härmed är översikten av de båda standard monitorsystemen avslutad och vi övergår till något som benämnes System Generation Program.

System Generation Program tillåter användaren av S-2 att generera en BCM eller en RBM som är skräddarsydd till hans anläggning och hans behov. Genereringen sker i två etapper. Under etapp 1 utväljes lämpliga programmoduler från BCM eller RBM som skrives på en anläggnings-huvudtape (installation master tape). Denna etapp (1) kontrolleras av kontrollkort. Under etapp 2 förvandlas anläggnings-huvudtapen till ett självladdande system som befinner sig på antingen skivminnet eller bandminnet eller på båda.

De två etapperna är oberoende av varandra och behöver ej utföras tillsammans. Output från etapp 1 är en relokerbar binär remsa i samma format som dess input, d.v.s. etapp 1 kan utföras flera gånger (om så önskas)innan etapp 2 genomföres. Under etapp 2 beskriver användaren det I/O-organ hans system har, storleken på papperet han vill använda i sin skrivmaskin eller radskrivare, antalet ggr systemet skall pröva ett I/O-organ om ett fel inträffat etc.

S-2 erbjuder sina användare två stycken assemblerspråk, Symbol (basic) och Extended Symbol. Symbol är anpassad att arbeta med den minsta maskinkonfigurationen. Extended Symbol är en utvidgning av Symbol och innefattar även möjlighet till PROCedure (t.ex. makro, se ref. nr 24, sid. 8 ff). Dessutom finns ett program för översättning från Extended Symbol till Symbol, vilket kan vara till nytta vid assemblering av elaborerade program på små maskiner.

Symbol finns tillgänglig i två utföranden, nämligen en one-pass och en two-pass version. One-pass versionen är bekvämast att använda ty källprogrammet behöver här endast läsas in en gång. I denna version uppfylles referenser framåt i programmet vid laddningstillfället. One-pass versionen innehåller inga restriktioner i språket.

Two-pass versionen medför att en mera lättläst objektkodlista erhålles och att referenser framåt i programmet göres vid tidpunkten för assemblering.

Sålunda är two-pass versionen lämplig vid programdokumentation och i vissa fall producerar den även ett effektivare objektprogram. I two-pass versionen kan andra etappen komma från ett yttre minne (RAD eller magnetband) eller genom en andra läsning av källprogrammet. S-2's assembleringsspråk tillåter programmeraren att ignorera adresseringsmoden och koda som om hela kärnminnet vore direkt adresserbart. För de flesta instruktioner väljer assemblern automatiskt en av de tre adresseringsmoderna (direkt, indirekt eller relativ adressering).

Extended Symbol är en multiple-pass assembler. Den arbetar på samma sätt som two-pass versionen av Symbol och skrives på multiple-pass from huvudsakligen för att spara tillhörande kärnminnesutrymme. Detta sker genom överföring till och från skivminnet.

Vi presenterar här nedan något som man i manualerna benämner Concordance Program (CP). Detta är användbart vid debugging och dokumentationsprogram. Indata till CP är ett källprogram i Symbol eller Extended Symbol. Utdata består av en tabell med alla använda symboliska namn, arrangerade i alfabetisk ordning och numret på varje rad där de använts. Kontrollkort tillåter användaren att tala om för CP att inte lista vissa symboler eller att lista endast speciella symboler som uppenbarar sig i särskilda fält.

S-2 systemet accepterar två fortranspråk, nämligen GEC FORTRAN IV (GF IV) och GEC Basic FORTRAN (GBF).

GF IV är definierat så att det inkluderar GBF som ett delspråk. Båda språken inkluderar Fortran IV-modellsatser. GF IV är höggradigt kompatibel med andra fortranspråk. De flesta program skrivna för andra FORTRAN-kompilatorer (både F-II och FIV) kan köras utan modifieringar med S-2 FORTRAN IV. Detta inkluderar följande språk som undergrupper:

ASA FORTRAN IV
IBM 7094 FORTRAN IV (Version 12)
ASA Basic FORTRAN
GEC Series 90 FORTRAN
GEC 90-2 FORTRAN IV

Dessutom accepterar den de flesta program skrivna i:

IBM 7090 FORTRAN II
IBM 1620 FORTRAN II
GEC Series 90 FORTRAN IV
GEC S-Seven FORTRAN IV
IBM Operating System/360 FORTRAN IV (H level)

Basic FORTRAN är avsett att köras på mindre maskinkonfigurationer. Det kan arbeta i antingen stand-alone mode eller under kontroll av Basic Control Monitor (BCM) utan tillgång till RAD-minne. S-2 FORTRAN inkluderar ASA Basic FORTRAN som en undergrupp och accepterar med minimal modifiering program skrivna i:

IBM 1620 FORTRAN II
IBM 709/7090 FORTRAN II
GEC Series 90 FORTRAN II

Till slut nöjer vi oss med att konstatera att det finns ett bibliotek, skrivet i Symbol, bestående av matematiska standardfunktioner av varierande slag.

Schematisk jämförelse av maskinernas registeruppsättning

UAC 1601	bits		CDC 1700	bits
Ackumulatorreg.	16		Adder/Shifter	16
Ackumulatorregistrets förlängn.	16		Arithmetic Reg.	16
Datareg.	16		Aux. Arithmetic Reg.	16
Skiftreg.	4		Program Address Reg.	15
Testreg.	2		Exchange Reg.	16
Prioritetsreg.	16		Address Reg.	16
Maskreg.	16		Function Reg.	8
Nivåreg.	4		Storage Data Reg.	18
Nivåbuffert reg.	4		Storage Address Reg.	15
Instruktionsreg.	16		Mask Register (4 bitar i "basic", med 1705 Interrupt Data Channel (16 bitar)	4 eller 16
Anm. De 16 allmänna registren (se dokumentation) och instruktionsregistret är alltid placerade i snabbminnet				

IBM 1800	bits	GE/PAC 4000	bits
Storage Address Reg.	16	Accumulator	24
Instruction Reg.	16	Auxiliary Accumulator	24
Storage Buffer Reg.	16	Place Counter AU 1 AU 2	14 16
Arithmetic Factor Reg.	16	Instruction Reg. AU 1 AU 2	24 26
Accumulator	16	Memory Buffer	24
Accumulator Extension	16	Counter for use with logic commands	5
Shift Control Counter OP Reg.	5	Index words, core memory locations 1-7	-
Index Reg.	16		
Auxiliary Reg.	16		
Channel Address Reg.	16		
		Märk 1. Auxiliary acc. är för GE/PAC 4040 realiserad med ett ord i kärnminnet	
		Märk 2. Det finns två aritmetiska enheter att välja mellan i 4000-systemet, AU 1 och AU 2. AU 1 är av serietyp ("basic" i 4040). AU 2 är av parallelltyp ("basic" i 4050 och 4060)	

PDP 8 (basic)	bits	PDP 9	bits
Accumulator	12	Accumulator	18
Link (carry reg.)	1	Link	1
Program Counter	12	Arithmetic Reg.	19
Memory Address Reg.	12	Control Reg.	
Switch Reg.	12	Multiplier-Quotient Reg. (optional)	18
Memory Buffer Reg.	12	Program Counter	15
Instruction Reg.	3	Instruction Reg.	5
		Memory Buffer Reg.	18
		Adder	19

GEC - S 2	bits
2 st Index Reg.	16
Link Address Reg.	16
Accumulator	16
Extended Accumulator	16
16 st Optional System Protection Reg.	16
8 st I/O Channel Reg.	16
Program Address Reg.	16
Zero	16
Temporary Storage Reg.	16
Memory Address Reg.	16
Control Reg.	16
Memory Data Reg.	16
Adder	16

Schematisk jämförelse med avseende på vissa systemfunktioner

SYSTEMFUNKTIONER	CDC 1700
Word length (bits)	18
Hardware priority interrupt	yes
Interrupt levels (number)	2 (16 extended)
Hardware multiply	yes
Core memory (No. of words)	4,12,16,24 or 32 K
Core memory cycle time	1.1 µs
Bulk memory (No. of words)	"fixed-head device" 65 to 8000 K disc memory 4800 K
High speed memory access time	drum memory: 8 ms (average)
Checking (parity or dual circuit)	parity
Direct memory access	yes
Cycle-Steal; max. delay	yes
Number of instructions	73
Indirect addressing	yes (several levels)
Relative addressing	yes
Indexing	yes
Part of core memory dir.addressable	block of 256 words
Double precision	yes
Floating-point (hardware)	no
Floating-point (software)	yes
Real-time clock	no
Limit timer (hardware)	yes
Limit timer (software)	yes
Circuit technique	discrete components

För att få en överskådlig bild av denna bilaga tejpas efterföljande sidor lämpligen ihop till ett dragspel. Ett enklare sätt, men ej så överskådligt, är att endast tejpa sista sidan som utviksblad på den näst sista i bilagan.

PDP-8	PDP-9
12	18
yes	yes
1	8
yes (optional)	yes (with extended elements)
4 K	8 or 32 K
1,5 µs	1,0 µs
disc 128K, serial drum 256K	drum memory: 32, 65 and 131 K
20 ms (disc memory)	15-20 ms
parity (optional)	parity (optional)
yes	yes
yes	yes
8 (basic)	16 (basic)
yes	yes
no	no
no	yes (auto-indexing)
256 words	8 K
yes (software)	yes (software)
no	no
yes	yes
yes (50 Hz)	yes
no	yes
yes	yes
hybrid silicon circuits	micro circuits

GEC - S 2	UAC 1601
16	16
yes	yes
132	16
yes	yes
Integral and/or external memories may be combined for capacities of 4K-64K words	8, 16 or 32 K
0.9 µs	ca 6 µs
see core memory	
disc: 17 to 25 ms	
parity	parity
yes (to ext.memory, optional)	no
yes	no
37	65 (22 operations)
yes	yes
yes	no
yes	yes
1 K	all
yes (hardware)	yes (software)
no	no
yes	yes
yes	yes
yes (optional)	no
yes	yes
monolithic integrated circuits	discrete components

SIEMENS 302	SIEMENS 303
24	24
yes	yes
2	2
no	no
8 or 16 K	4, 8, 12 or 16 K
1.5 µs	33 µs
drum memory: 65, 131 or 262 K	drum memory: 65, 131 or 262 K
31 ms (average)	31 ms (average)
parity	parity
yes	yes
yes	yes
23	31
yes	yes
no	no
no	no
all	all
yes (software)	yes (software)
no	no
no	no
yes	yes
yes	yes
yes	yes
monolithic technique with changeable circuit cards	cilicon technique with changeable circuit cards

SIEMENS 304	SIEMENS 305
24	24
yes	yes
2	2
yes	yes
8 or 16 K	8 or 16 K
1.5 µs	1.5 µs
drum memory 65, 131 or 262 K ext. memory: 16K; disc memory 1800 K disc memory: 12 to 170 ms core memory: 3 µs, drum memory 31 ms parity	→ (drum memory 65, 131 or 262 K; ext. memory: 16K; disc memory 1800 K) → (disc memory: 12 to 170 ms; core memory: 3 µs, drum memory 31 ms) parity
yes	yes
yes	yes
40	46
yes	yes
no	no
no	no
all	all
yes (software)	yes (software)
no	yes
no	
yes	yes
yes	yes
yes	yes
cilicon technique with changeable circuit cards	monolithic technique with changeable circuit cards

ARCH 102	ARCH 9000	ARCH 2020
13	18	24
yes	yes	yes
4	4	3
yes	yes	yes
4 or 8 K	8K or multiples of 8Kto65K	6µs:8,16,24 or 32 K 2µs:16 or 32 K
6 µs	6 µs	6 µs
parity	parity	parity
no	no	no
16	23	300 + "extra code"
yes	yes	yes
no	no	no
no	no	yes
256 words	8 K	all
yes (software)	yes (software)	yes (hardware)
no	no	no
no	yes	yes
yes	yes	yes
yes	yes	yes
yes	yes	yes
discrete components silicon technique	discrete components silicon technique	discrete components silicon technique

SYSTEMFUNKTIONER
Word length (bits)
Hardware priority interrupt
Interrupt levels (number)
Hardware multiply
Core memory (No. of words)
Core memory cycle time
Bulk memory (No. of words)
High speed memory access time
Checking (parity or dual circuit)
Direct memory access
Cycle-Steal; max delay
Number of instructions
Indirect addressing
Relative addressing
Indexing
Part of core memory dir.addressable
Double precision
Floating-point (hardware)
Floating-point (software)
Real-time clock
Limit timer (hardware)
Limit timer (software)
Circuit technique

GE/PACK 4000	IBM 1800
24	18
yes	yes
8 (128 extended)	12 (24 extended)
yes (4050 and 4060) no (4040)	yes
1 to 16 K (4040) 4 to 65 K (4050 and 4060)	4, 8, 16 or 32 K
1,7,3.4 or 5 µs for 4050 and 4060 2.38 or 4.08 µs for memory 16K;5µs(4040)	2 or 4 µs
16 K to 262 K in steps of 8 K	256 K
15 - 20 ms	ca 30 ms
parity	parity
yes	yes
yes	yes; 2,25 or 4,5 µs
100	31
yes (4050 and 4060) no (4040)	yes
yes	yes
yes	yes
16 K	256 words
yes (4050 and 4060) no (4040)	yes
yes (4050 and 4060)	no
yes (4040)	yes
yes	yes
no	yes
yes	yes
discrete components	discrete components

Jämförelse med avseende på interna hastigheter med hjälp av instruktionsblandningar (Mixar)

Instruktionsblandningar (mixar) användes vid jämförelse mellan olika maskiners interna snabbhet. Dessa blandningar grundar sig på kännedom om användningsfrekvensen av olika instruktioner. Instruktionsfrekvensen för olika program gör att en vägd medeloperationstid kan beräknas. Instruktionerna indelas i fyra huvudklasser:

1. Flytande aritmetik
2. Multiplikation, division (fix talrepresentation)
3. Hoppoperationer (och operationer med högra hastighet än 4)
4. Operationer av samma snabbhet som addition (fix talrepresentation)

Efterföljande tabelluppställning anger dels en "blandning" som konstrueras av Raichelsen och Collins gällande för beräkningar av administrativ karaktär vid reeltidssystem, dels en instruktionsblandning för beräkningar av "vetenskaplig" karaktär angiven av Gibson.

Program av teknisk-matematisk natur (enl. Gibson)	Frekvens f_{1i}%	Program vid reellt tidsappl. (enl. Raichelsen, Collins)	Frekvens f_{2i}%
Klass 1		Klass 1	
Flytande addition	6,9	Flytande addition	0,5
Flytande multiplikation	3,8	Flytande multiplikation/division	0,2
Flytande division	1,5		
F_{11} =	12,2	F_{21} =	0,7
Klass 2		Klass 2	
Fix multiplikation	0,6	Fix multiplikation och division	0,3
Fix division	0,2		
F_{12} =	0,8	F_{22} =	0,3
Klass 3		Klass 3	
Hopp	16,6	Ovillkorliga hopp	5,0
Instr. utan ref. till minne	5,3	Villkorliga hopp	7,5
Indexering	18,0	Index-hopp mm	3,5
		Instr. utan ref. till minne	2,5
F_{13} =	39,9	F_{23} =	18,5
Klass 4		Klass 4	
Addition/subtraktion(fix)	6,1	Addition/subtraktion (fix)	8,0
Överföringsinstr.	31,2	Öka minnet med I	3,0
Sökning och jämf.	3,8	Överföring	47,5
Skift	4,4	Logiska instr.	7,0
Logiska op.	1,6	Skift	7,5
		Jämförelse	4,0
		Diverse	3,5
F_{14} =	47,1	F_{24} =	80,5

Klassfrekvenserna F_{ij} är då summan av instruktionsfrekvenserna f_{ij} i respektive klass. En vägd medeloperationstid för en viss applikationstyp i kan beräknas enligt

$$E_{oi} = \beta \cdot 10^{-2} \cdot \sum_{j} f_{ij} \cdot e_j$$

där f_{ij} är frekvensen för instruktionen j som har operationstiden e_j och β är en konstant som beror på instruktionslogiken.

Beräkning av datamaskiners interna hastighet enligt de nyss angivna principerna har i många fall visat sig ge en god överensstämmelse med faktiska försök (tidtagning av rent interna processer utan transporter). I de fall överensstämmelsen har varit sämre, har det i regel berott på att man ej tagit hänsyn till faktorn β som utjämnar olikheterna i instruktionslogiken. Några säkra värden på β kan ej anges då en tillräcklig mängd undersökningsmaterial saknas.

Tills vidare anges några ungefärliga värden på β

En adress logik	β = 1,00
En adress logik med multippla (ca 10) ackumulatorregister	β = 0,75 - 0,80
Två adress logik	β = 0,75
Tre adress logik	β = 0,60 - 0,65

Metoden får dock enligt vår mening betraktas som ganska osäker. Detta bl.a. beroende på svårigheten att klassificera de olika instruktionerna. Även metoden att ta medeltalet av antalet varianter av en viss instruktion (t.ex. olika slags skift) förefaller vansklig då en viss variant kan vara speciellt mycket använd medan andra däremot användes mindre ofta. Observera också till sist att den interna processtiden ej får förväxlas med programmets totala behandlingstid.

Nedan följer några exempel på uträkning av operationstider enl. Raichelsen, Collins

UAC 1601

Klass 1	0,5	Flytande addition	= 2000
	0,2	Flytande multiplikation/ division	= 5000

$$\sum_j f_{1j} e_j = 2000$$

Klass 2	0,3	Fix multiplikation och division	= 96,3

$$\sum_j f_{2j} e_j = 28,9$$

Klass 3	5,0	Ovillkorliga hopp	= 52,4
	7,5	Villkorliga hopp	= 33,7
	3,5	Index-hopp mm	= 59
		Instr. utan ref. till minne	= 36

$$\sum_j f_{3j} e_j = 811,3$$

Klass 4	8,0	Addition/subtraktion (fix)	= 42,7
	3,0	Öka minnet med 1 (=kortaste additionstid)	= 42,7
	47,5	Överföring (Load, Store)	= 44,0
	7,0	Logiska instr.	= 45,2
	7,5	Skift	= 34,4
	4,0	Jämförelse	= 41,9
	3,5	Diverse (I/O)	= 42,0

$$\sum_j f_{4j} e_j = 3448,7$$

$\beta = 0,75$

Medelop.tid = 48,4 µs

GE/PAC 4040

Klass 1	0,5	Flytande addition	= 990
	0,2	Flytande multiplikation/ division	= 2046
	$\sum_j f_{1j} e_j$ = 1518	Anm. Dubbel precision ej medtagen	
Klass 2	0,3	Fix multiplikation och division	= 2822
	$\sum_j f_{2j} e_j$ = 847	Anm. Multiplikation av tal bestående av 12 bitar, svaret i 24 bitar	
Klass 3	5,0	Ovillkorliga hopp	= 14
	7,5	Villkorliga hopp	= 14
	3,5	Index-hopp mm	= 20
	2,5	Instr. utan ref. till minne	= -
	$\sum_j f_{3j} e_j$ = 245		
Klass 4	8,0	Addition/subtraktion (fix)	= 79
	3,0	Öka minnet med 1	= 16
	47,5	Överföring	= 61
	7,0	Logiska instr.	= 16
	7,5	Skift(ej dubbelskift)	= 325
	4,0	Jämförelse	= 141
	3,5	Diverse	= -
	$\sum_j f_{4j} e_j$ = 6692		

β = 1

Medelop.tid = 93,0 µs

GE/PAC 4060

Klass 1	0,5	Flytande addition	= 92
	0,2	Flytande multiplikation/ division	= 250

$$\sum_j f_{1j} e_j = 171$$

Klass 2	0,3	Fix multiplikation och division 24 x 24 mult.	= 219

$$\sum_j f_{2j} e_j = 66$$

Klass 3	5,0	Ovillkorliga hopp	= 1,7
	7,5	Villkorliga hopp	= 1,7
	3,5	Index-hopp mm	= 3,4
	2,5	Instr.utan ref, till minne	= -

$$\sum_j f_{3j} e_j = 33$$

Klass 4	8,0	Addition/subtraktion	= 1,7
	3,0	Öka minnet med 1 (kortaste additionstid)	= 1,7
	47,5	Överföring (Load, Store and Move)	= 7,9
	7,0	Logiska instr.	= 3,4
	7,5	Skift (ej dubbelskift)	=10,0
	4,0	Jämförelse (eller subtract)	= 1,7
	3,5	Diverse	= - -

$$\sum_j f_{4j} e_j = 500$$

$$\beta = 1$$

Medelop,tid = 7,7 µs

Schematisk jämförelse av räknetider

SYSTEM	CDC 1700	GE/PAC 4020	4040	4050 I	4050 II	4060	IBM 1800		PDP-8	PDP-9	GEC s-2
Cykeltid µs	1.1	1.6	5	4.1-5.1	3.4-4.1	1.7-2.4	4	2	1.5	1.0	0.9
A + B fix, EP µs	2.2	3.2	16	10.2	6.8	3.4	9	4.5	3	2.0	2.25
A x B fix, EP µs	7.0	12	300	22.7	19.3	15.9	30.5	15-45	21	3.5	10.35
A + B fix, DP µs							13.5	7	16.5		
A x B fix, DP µs									307		
A + B flytande (minst 24 bits) µs	150	222	990	246.8	161.5	92.4	520	260	405	70	
A x B flytande (minst 24 bits) µs	250	172	1460	271.3	189.0	109.4	640	320	530	320	
A/B flytande (minst 24 bits) µs	450	244	2631	335.6	233.0	140.2	860	430	590	370	
A/D-omvandlare antal bitar/omvandlingshast. µs	12/15	12/~20	12/~20	12/~20	12/~20	12/~20	8,11,14/ 29,36,44		12/35	12/35	15/~20
Multiplexer, punkter per sek.	50K	40K	40K	40K	40K	40K	100K		25K	50K	40K

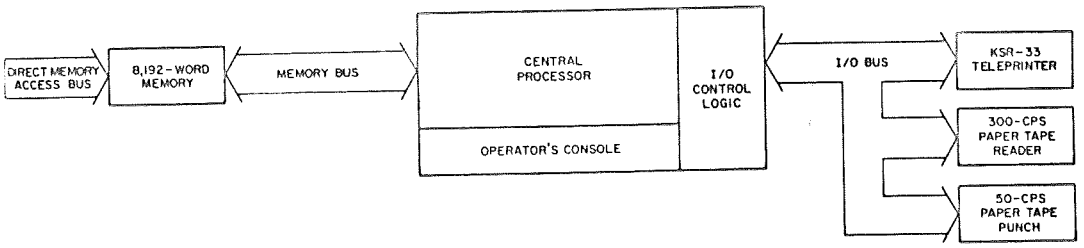


Figure 1 Basic PDP-9

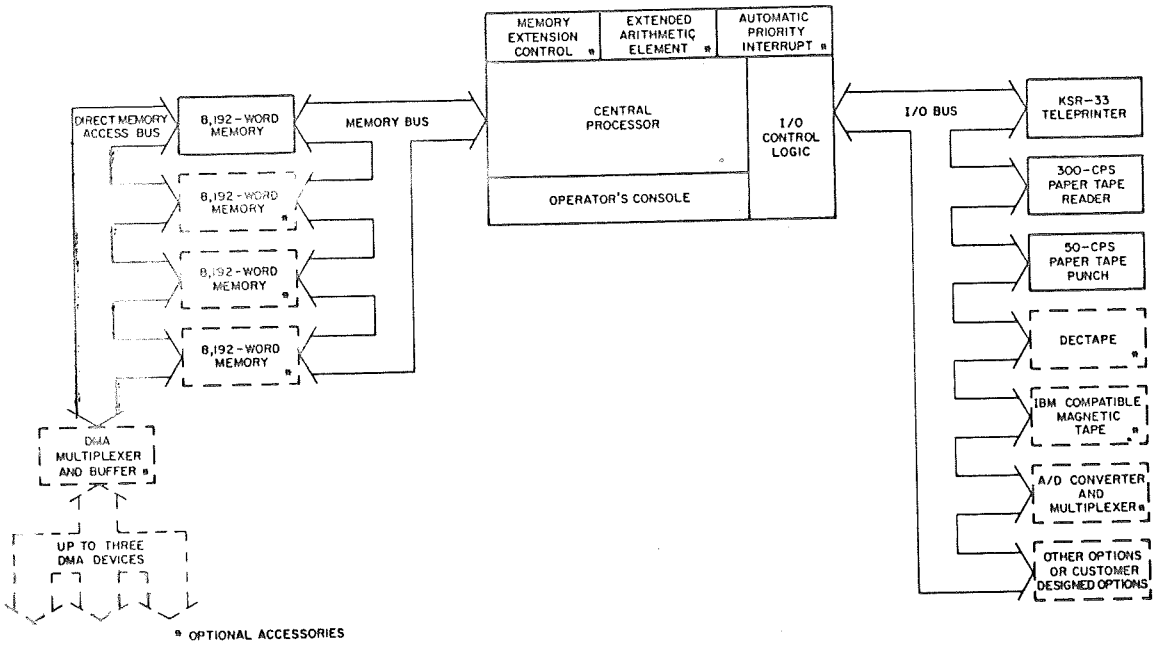


Figure 2 Expanded System Configuration

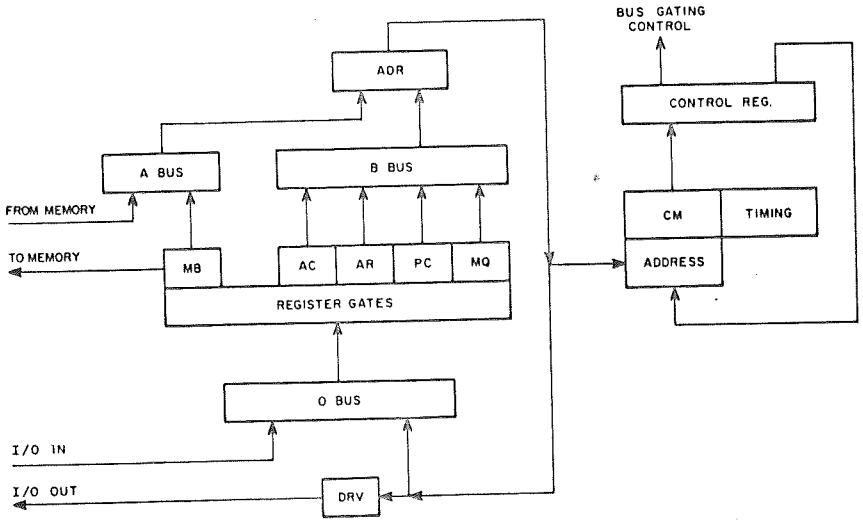


Figure 3 Central Processor Organization

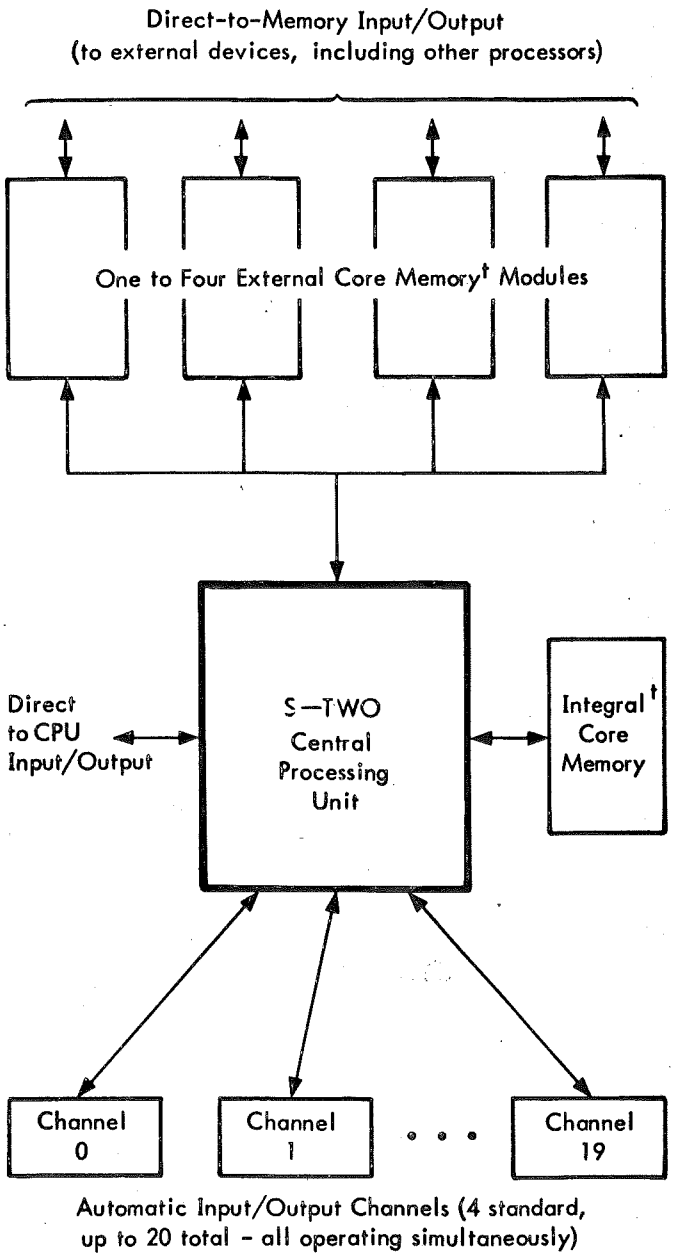
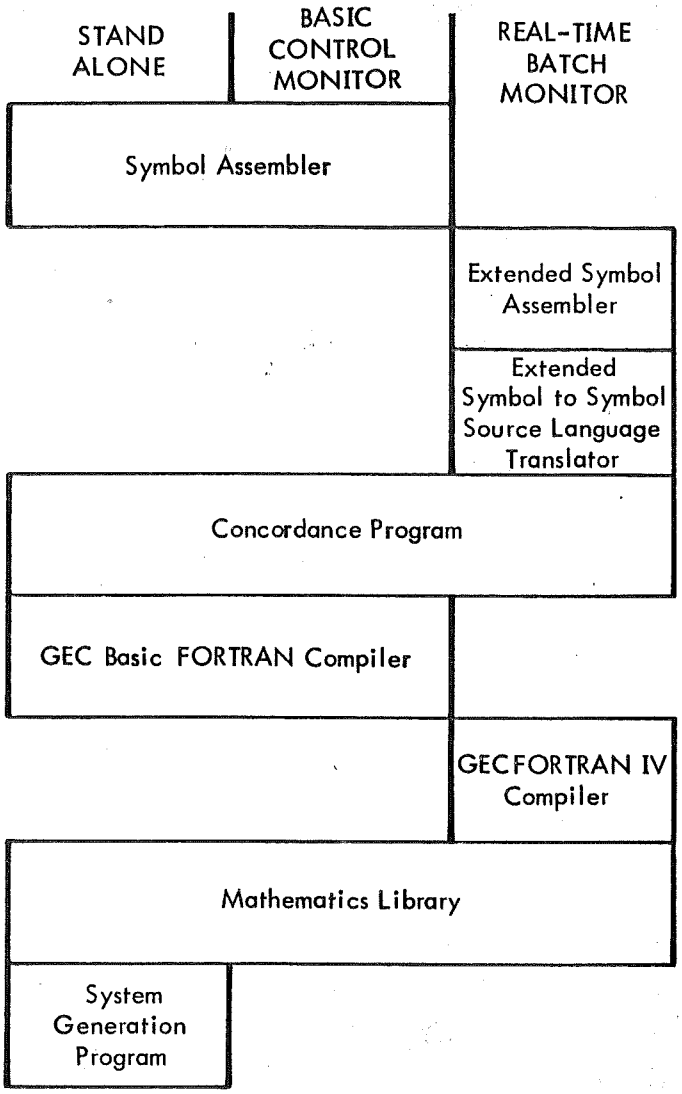


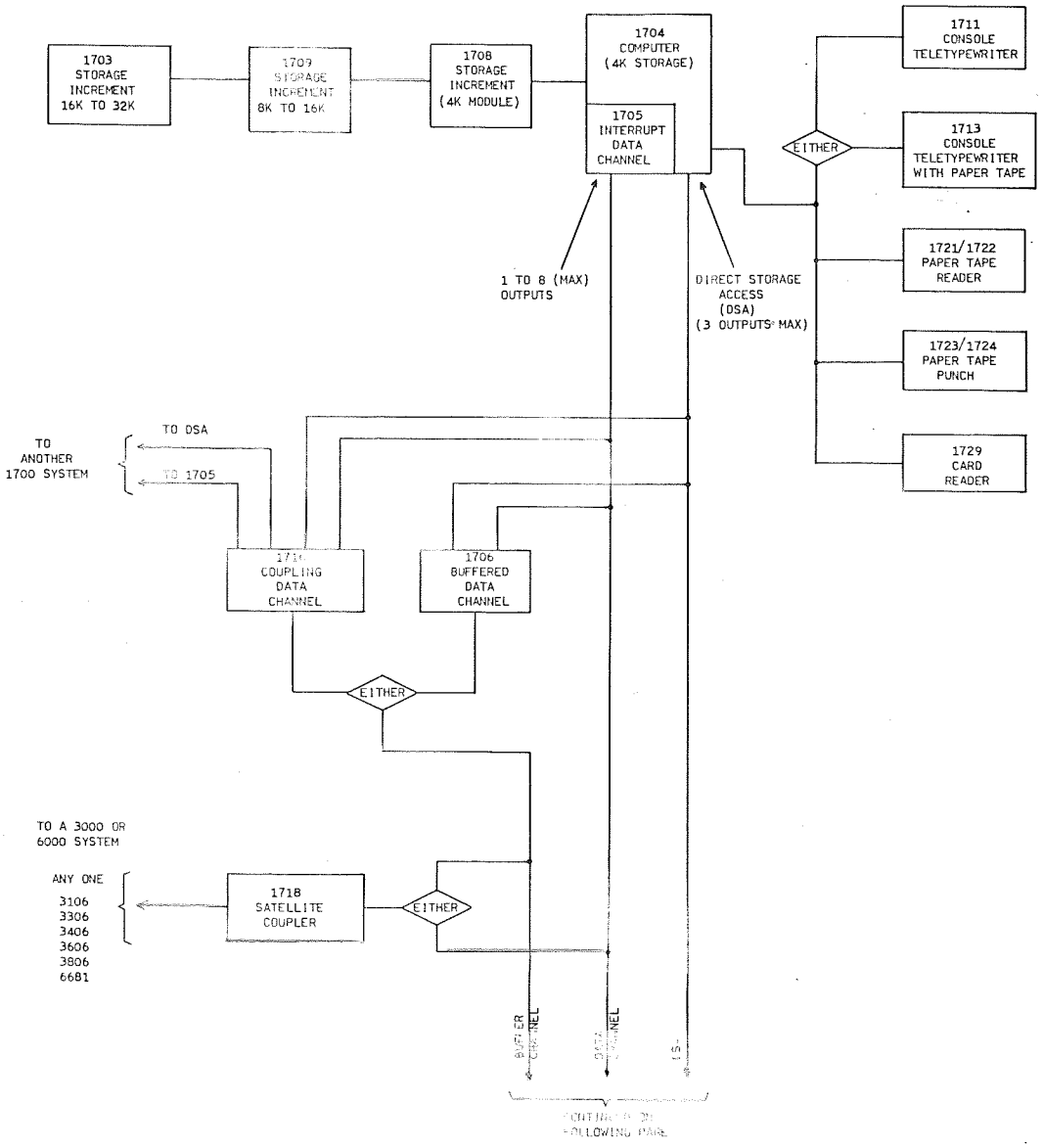
Figure 1. S—TWO System Configuration



S-TWO programming systems

†Integral and/or external memories may be combined for capacities of 4K-64K words.

1700 COMPUTER SYSTEM CONFIGURATION



Ordlista

Accesstid	Den tid som det tar att lokalisera en viss plats i ett minne och därur läsa eller däri skriva in information
Batch processing	Satsvis bearbetning; data samlas på hög under viss tidsperiod för att sedan bearbetas på en gång
Buss	T.ex. databuss. Man har här på ett praktiskt sätt löst problemet att förbinda en mängd enheter, som på ett dynamiskt sätt skall kunna kommunicera med varandra. Man har en central ledning (bussen) med vilken samtliga enheter är förbundna. Ett kontrollsystem reglerar sedan kontakterna från de enskilda enheterna till bussen, så att önskade förbindelser uppkopplas. (Buss = highway i en del maskiner)
CPU	Centralenhet (eng. Central Process Unit);innehåller styrenhet, aritmetisk enhet och primärminne. Hos flera fabrikanter räknas ej primärminnet in i centralenheten
DBS	Databehandlingssystem; består av DMS eller PDMS plus tillhörande programvara (se DMS resp. PDMS)
DMS	Datamaskinsystem (eng. Compute System); den utrustning som utför databehandling
Optional	Utrustning utöver basic
PDMS	Processdatamaskinsystem; databehandlingssystem för kontroll, övervakning, logging och reglering av industriella processer i reell tid (exkl. programvara)
Relokerbar	Startadressen för programmet kan väljas. Så är ej fallet för program i absolut form
Timesharing	Här råder stor begreppsförvirring och många sätter likhetstecken mellan timesharing och multiprogrammering. Vi vill göra följande uppdelning
Tiduppdelad I/O-matning	Då ett program bearbetas i CPV kan samtidigt I/O-matningar företagas
Tiddelning	Avbrott i ett pågående program sker då anmodan om bearbetning av ett program med högre prioritet görs. När det högprioriterade programmet inte längre tar CPV i anspråk, återgår maskinen till det avbrutna programmet

Multiprogrammering	Flera program kan uppta plats i kärnminnet och bearbetas samtidigt.

För övrigt hänvisas till ref. 14 och 28.

Om en dataanläggning i Kinne
sa en butter och misstänksam finne:
Ingen plåtmakapär,
lär sig räkna så där.
Nä, det sitter nån jäkel därinne.

Starta trumman, vrid på nyckeln
Åk iväg på minnescykeln

A	ADRESS	Adress betecknar platsen inne i maskinens goda minne
	ARITMETIK (serietyp)	Jag är långsam knogar villigt köpes för att det är billigt
	ARITMETIK (parallelltyp)	Parallellaritmetik, snabb och bra om Du är rik
	AVAILABILITY	AVAILABLE är grej som går DOWNTIME är det när den står
B	BIT	Binär siffra är så liten noll och ett blott finns i biten
	BUFFERTMINNE	Buffertminnet möjliggör att data väntar utanför
C	COMPATIBEL	Compatibla är compjuttrar om på samma språk de muttrar
D	DATA	Data är information behandlade med instruktion
	DDC	DDC kan utan vådor remplacera PI-lådor
	DEBUG	Debug är att felen rätta DUMPA är att minnet "tvätta"
	DIREKT ACCESS	Direkt access är snabbmetod hämta eller lagra ord
F	FLÖDESSCHEMA	Flödesschemat måste göras innan kodning kan utföras
I	IMPLEMENTERA	Färdig med att programmera? Sätt igång, implementera!
	INSTRUKTION	Instruktionen informerar hur maskinen opererar

L	LÄSARE	Läsar´n sväljer råd och data med kort och remsa man kan den mata
M	MONITOR	Monitorn är softwaredelen som ger norrsken på panelen
	MONOLITISK	Monolitisk är en krets som tillverkats genom ets
	MTBF (Mean Time Between Failure)	MTBF talar om hur ofta servicemannen kom
	MULTIPLEXER	Multiplexa är att scanna cykliskt utan nånsin stanna
O	ON-LINE	On-line är då man är inne och kan snacka med ett minne
R	REGISTER	Om Du dataordet mister sök det uti ett register
S	SKIFT	Ställ små bitar i en rad Ett steg åt sidan sedan tag
	STANS	Stansen golfarn´s dröm besannar hål i ett tills remsan stannar
	STYRENHET	Styrenhet är komponenten fungerande som dirigenten
Å Ä Ö		Vår maskin går bet när den ska skriva Å, Ä, Ö på svenska

Referenslista

Ref. nr 1 Bubenko Jr. Janis, Databehandlingsteknik. Studentlitteratur Lund 1967

2 Control Data 1700 Compute System: Analog/Digital Systems. General Information Manual

3 Control Engineering Sept. 1966

4 Control Engineering March 1967

5 Control Engineering April 1967

6 Datamation Vol. 12 No. 2 Febr. 1966

7 Dopping Olle, Datamaskiner och databehandling. Statskontoret/Studentlitteratur Lund 1966

8 Fröberg och Sigurd, Datamaskiner. Lund 1965

9 Industria okt. 1967

10 Modern Datateknik nr 9 1967

11 Schoeffler J D, se ref. nr 6

12 Scientific America. Sept 1966

13 Teknisk Tidskrift nr 34 1967

14 IFIP-ICC Vocabulary of Information Processing. Amsterdam 1966

15 Control Data 1700 Computer System, System Manual

16 GE/PAC 4000 GET-3201 B, Systems Manual

17 IBM 1800 File No. 1800-01 Form A26-5918-1, Data Acquision and Control System Reference Manual

18 PDP-8 Small Computer Handbook, Digital Equipment Corporation. Maynard Mass.

19 PDP-9 F-91 Digital Computer Spec.

20 S 2 S-Two Computer Reference Manual

21 Intern dokumentation L M Eriksson

22 GE/PAC 4000 PCP-102B, Programming Manual

23 PDP-9 DEC-9A-BSAA-D, Software Summary

24 S-Two Programming Systems, General Information

25 CDC 1700 Operating System, Reference Manual

26 IBM 1800 Form A26-5920-2 Data Acquisition and Control System, System Summary

27 M H Aronson, M L Klein och H C Morgan, Digital Techniques for Computation and Control. Instruments Publishing Co. Pittsburgh

28 SEN 0116 Datamaskiner, ordlista. Sveriges Standardiseringskommission

Ytterligare litteratur i ämnet kan med fördel sökas i referensregistren i ovanstående publikationer, som ju idag är av färskt datum. Industria, nov. 1967, innehåller en god översikt av tillgänglig litteratur inom området administrativ databehandling. Eftersom denna översikt täcker ett mycket brett fält, finns det här även material som är av intresse för vårt arbete. I tidskriften Control Engineering finns årligen en sammanställning av samtliga installerade processdatamaskiner. Sveriges Standardiseringskommission håller f.n. på med att utarbeta en fullständigare ordlista för datamaskinområdet.