HARDWARE ARKITEKTUR

for

SPC/2 Datamaten

Marts 1982

Knud Arne Nielsen.

Indholdsfortegnelse

1. Indledning.

Denne rapport er en status rapport for SPC/2´s hardwarearkitektur. Den kan ikke opfattes som en design specifikation for maskinen, da forskellige dele endnu kun foreligger som skitser.

2. Oversigtsbeskrivelse.

En SPC/2 er et multiprocessor system bestående af et antal CPU´er og et antal intelligente IO kontrollere. CPU´er og kontrollere kommunikerer over en bus kaldet fællesbussen. CPU´erne er af typen MC68000. De er forsynet med lokalt lager på indtil 16 Mb. En CPU er forsynet med en Memory Management Unit, MMU, som varetager lagerbeskyttelse og letter administrationen af lageret. En CPU kan læse/skrive i sit eget lager uden at accesse fællesbussen, hvorfor lageret betegnes lokalt. De intelligente kontrollere varetager input/output for systemet. Deres opgave er at styre de ydre enheder og at overføre data mellem de ydre enheder og CPU´ernes lokale lager.

De intelligente kontrollere realiseres med 8085 og periferikredse kendt fra SPC/1.

Der kan maksimalt tilsluttes 16 enheder til fællesbussen, hvilket betyder, at antallet af CPU´er og antallet af kontrollere maksimalt kan blive 16. En intelligent kontroller inklusiv dens nødvendige lokale lager består af et enkelt printkort. En CPU eksklusiv dens lokale lager søges også realiseret på et enkelt printkort, mens det lokale lager til en CPU findes på et eller flere printkort med 1/2 Mb pr. kort.

Det maksimale antal printkort i en maskine begrænses til 24, hvilket medfører, at en maskine f.eks. kan bestå af 8 CPU´er, 8 kontrollere samt 8 lagerkort. Det forventes, at lagerkortenes kapacitet forøges væsentligt i fremtiden.

Fordelene ved en struktur som denne er flere:

En CPU accesser stort set kun sit eget lokale lager, idet fællesbussen kun anvendes til input output og meddelelser mellem CPU´erne. CPU´erne forsinker ikke hinanden, når de accesser deres eget lager. Maskinens kapacitet, hastighed, vil derfor være proportional med antallet af CPU´er, indtil hastigheden begrænses af input output over fællesbussen.

En maskine med flere CPU´er er redundant. Dette åbner mulighed for ved udbygning af programmel og materiel at gøre maskinen ekstrem pålidelig.

3. Fællesbussen.

Fællesbussen forbinder indtil 16 enheder. Fællesbussen består af en databus på 16 bit, D(15:0), en adressebus på 34 bit, A(33:0), en mekanisme til prioritering af adgangen til fællesbussen samt diverse styresignaler.

Bussen er en demand multiplekset bus, som giver adgang til et meget stort lager. Dette lager er f. eks. lokalt lager i enheder tilsluttet fællesbussen. Bussen kan kun udføre to bassale funktioner, læsning og skrivning samt en funktion afledt heraf: READ MODIFY WRITE, RMW. Læsning eller skrivning sker i den adresserede lagercelle i den adresserede enhed. Der findes ingen specielle funktioner som f.eks. læs input port, skriv output port, læs DMA, skriv DMA, send interrupt o.s.v.

Bussen kan udføre enhver funktion, som kan oversættes til en læsning eller skrivning i en adresseret lagercelle. Det er svært at forestille sig funktioner, som ikke kan oversættes til læsning eller skrivning i en adresseret lagercelle, når "lagercelle" opfattes passende bredt.

3.1. Adressering på fællesbussen.

Adressebussen består af en 34 bit byte adresse, A(33:0), som opfattes på følgende måde:

A(33:30) = UNIT(3:0)

De fire mest betydende bit af adressen adresserer en enhed tilsluttet bussen.

A(29:24) = ASN(5:0)

5 bit, der angiver et Address Space Number. Disse 5 bit anvendes først og fremmest ved adressering af MC68000´s lager fra en intelligent kontroller, d.v.s. når en kontroller læser eller skriver data i en CPU´s lager. ASN anvendes ved relokeringen og lagerbeskyttelsen og omtales nærmere ved beskrivelsen af MMU.

A(23:0)

En 24 bit byte adresse, som muliggør adressering af 16 Mb i den adresserede enheds lager.

Fællesbussen opnår sin store fleksibilitet ved, at kun UNIT(3:0), som adresserer en enhed, har en veldefineret betydning. Hvilken betydning, der lægges i de øvrige adressebit, A(29:0), afhænger af den adresserede enhed.

Når den adresserede enhed er en MC68000, vil den øvrige del af adressen normalt være et ASN og en lageradresse. Når den adresserede enhed er en kontroller, vil adressen normalt tolkes på en anden måde. F.eks. vil kun de 16 mindst betydende bit have betydning. Skrivning af en bestemt byte kan f.eks. betyde at

overføre et sektornummer til en disk access, læsning af en anden byte kan betyde læsning af status fra en ydre enhed o.s.v.

Betydning af læsning og skrivning i en kontrollers lager fastlægges ved konstruktion af enheden og beskrives i en "Programmers Manual" for den pågældende enhed.

3.2. Prioritering.

Da flere uafhængige enheder samtidigt kan forsøge at lave en access til fællesbussen, er det nødvendigt med en prioritering mellem enhederne. Prioriteringen kunne tænkes udført på flere måder, central/fordelt prioritering med fast/variabel/roterende prioritet.

Prioriteringen, som er valgt på fællesbussen, kan bedst karakteriseres som fordelt prioritering med roterende prioritet. Fordelt prioritering betyder, at der ikke findes en central busmaster, som foretager prioriteringen, og roterende prioritet betyder, at der ikke findes en enhed, som altid har højeste prioritet.

Prioriteringen sker ved hjælp af følgende signaler:

BR:

Bus Request, aktivt lavt signal, som forløber parallelt til alle enheder på bussen. Når en enhed ønsker bussen, sætter den BR lav. BR betyder altså at mindst en enhed ønsker bussen.

BGI, BGO:

Bus Grant In, Bus Grant Out, er et signal, som forløber cyk-

lisk gennem alle enheder. BGI er et input signal, som er forbundet til den foregående enheds BGO. BGO er et output signal fra en enhed. Signalet er forbundet til den følgende enheds BGI.

BGACK:

Bus Grant Acknowledge er et aktivt lavt signal, som forløber parallelt til alle enheder på bussen. Til ethvert tidspunkt er en af enhederne busmaster. BGACK angiver for busmasteren, at den ikke er busmaster mere.

Prioriteringen sker på følgende måde:

Når en enhed ønsker bussen, sætter den signal på BR. Når busmasteren modtager BR, og den ikke ønsker bussen mere, sætter busmasteren signal på BGO, som er forbundet til den følgende enheds BGI. En enhed, som ikke ønsker bussen, sætter signal på BGO, når den modtager BGI. En enhed, som ønsker bussen, sætter intet signal på BGO, når den modtager BGI. I stedet for sætter den signal på BGACK, hvorved busmasteren ser, at en ny enhed er selekteret. Busmasteren fjerner BGO, og den selekterede enhed er den nye busmaster.

En og kun en enhed skal selekteres som busmaster ved power up reset.

Ved realiseringen af ovenstående kan det vise sig praktisk at tilføje et par signaler ekstra.

Enhederne konstrueres i øvrigt sådan, at ingen enhed kan beslaglægge bussen i mere end en buscycle, hvis en anden enhed ønsker access til bussen. En enhed er altså ikke i stand til at forhindre andre enheder i at anvende bussen.

3.3. Fejlsituationer.

Der kan optræde to forskellige fejlsituationer:

1. En enhed tilknyttet fællesbussen overholder ikke spillereglerne på bussen. Der eksisterer altså en hård fejl i maskinen, som muligvis ikke kan omgås. Nogle af disse fejl vil dog kunne omgås ved, at en enhed overvåger kommunikationen på bussen og griber ind, når den detekterer en fejlsitustion. Situationen, hvor den adresserede enhed ikke svarer, vil kunne klares på denne måde.

2. Den adresserede enhed detekterer en fejl. Dette kan f.eks. være overtrædelse af lagerbeskyttelse eller fejl ved en lageraccess. Denne fejl meddeles enheden, som udfører accessen til fællesbussen, over en ledning i denne.

3.4. Bushastighed.

For at vurdere bushastigheden skal bussens funktion beskrives nøjere.

En enhed A ønsker at skrive i enhed B´s lager.

Enhed A´s adresse dekodes, og når dekodningslogikken genkender en access til fællesbussen, genereres WAIT i enhed A.

Businterfacen i enhed A sender BR og venter, til den får rådighed over bussen. Når dette sker sætter enhed A adressen ud på adressebussen, og først da adresseres enhed B.

Enhed B er i færd med en lager cycle, hvorfor accessen til B´s lager fra fællesbussen forsinkes, til B´s lager cycle er afsluttet. Når B´s egen lager cycle er afsluttet, genereres WAIT

i B, mens accessen til B´s lager foregår. Når data fra B´s lager er udlæst, løber de over databussen til enhed A, og hele cyclen er afsluttet. Herefter finder en ny prioritering sted, hvis der er Busrequest.

En buscycle består altså af følgende hændelser:

A. Prioritering og adgang til fællesbus.

B. Adgang til den adresserede enheds lager.

C. Lageraccess.

Ved vurdering af bushastigheden er der flere tal, der har interesse:

1. Middel access tid til anden enheds lager.
2. Maksimal access tid til anden enheds lager.
3. Bus båndbredde.

For at give et groft skøn over de tider, der kan forventes, antages:

Lager access/cycle tid: 0.5 us.
CPU cycle tid: 1.0 us.

Det vil være naturligt at lade dele af en buscycle forløbe parallelt med den foregående eller efterfølgende buscycle. Prioriteringen af den følgende buscycle skal f.eks. ske parallelt med den nuværende buscycle.

Den maksimale tid fællesbussen er optaget af en buscycle bliver derfor lig med summen af en CPU cycle og en lager cycle d.v.s. 1.5 us.

Maksimal bus cycle: 1.5 us.

Middel bus cyclen bliver derimod lig med en halv CPU cycle plus en lager cycle d.v.s 1 us. Se også Appendix 1.

Middel bus cycle: 1 us.

Busbåndbredde: 1 Mc/s svarende til 2 Mbytes/sek.

Maksimal access tid, når 16 enheder ønsker access til fællesbussen:

16 gange 1.5 us = 24 us

Middel access tid, når 16 enheder ønsker access til bussen

8 gange 1 us = 8 us

I den beskrevne bus sker to ting parallelt:

1. Prioritering.
2. Adressering og dataoverførsel.

Det er en forudsætning for beregningerne, at prioriteringen altid kan ske hurtigere end en cycle på fællesbussen. Da en minimum cycle på fællesbussen er lig med en lagercycle, 500 ns, kan denne forudsætning altid anses for opfyldt.

Hvis prioriteringen ikke sker parallelt med adresseringen og dataoverførselen, formindskes busbåndbredden, og access tiden til bussen forøges.

Hvis en buscycle kan opdeles i flere faser, som kan forløbe parallelt, kan busbåndbredden forøges yderligere.

Ovenstående beregninger skal anvendes med forsigtighed. Dels er tiderne for CPU cycle og lager cycle skønnede tider, dels er det vanskeligt i en given situation at afgøre, om middel tider, minimale tider eller maksimale tider er afgørende. Man kan dog med sikkerhed fastslå at busbåndbredden overstiger 1 Mbytes/sek.

3.5. Transport hastighed i SPC/1.

Datatransporter i SPC/1 sker fra disken til det faste lager med hjælp af DMA og dernæst fra det faste lager til brugerens lager ved hjælp af lager til lager DMA. Buffer størrelsen i det faste lager vil normalt begrænse transporthastigheden, så denne bliver fire sektorer pr. disk rotation.

Ved en access til en hard disk bliver transport hastigheden derfor 1kb/16.6 ms, svarende til 60 kb/sek.

Ved access til en floppy disk bliver transporthastigheden 6 kb/sek.

I de nuværende back up programmer sker data transporten med en hastighed på 16 us/byte svarende til 62 kb/sek.

Transport hastigheden på fællesbussen er 10 til 30 gange større end den hastighed, hvormed input/output sker i SPC/1.

4. Interrupt i SPC/2.

Flere processorer deler intelligente kontrollere, IOP. En IOP skal kunne sende interrupt til flere processorer på flere forskellige interrupt niveauer. En processor kan modtage interrupt på 7 forskellige niveauer fra f.eks. 8 forskellige IOP´er.

Det må være et krav til interrupt systemet, at et interrupt ikke giver anledning til interrupt i en processor, som ønskes uberørt.

Bussen kan forsynes med 7 interrupt request ledninger for hver processor i bussen f.eks. 8*7 ledninger. Når en processor akcepterer et interrupt på et givet niveau, må den undersøge alle IOP´er, der kan give interrupt på det pågældende niveau, for at finde interruptårsagen. Ved denne fremgangsmåde skal en processor indeholde 7 interrupt request ledninger, mens en kontroller, som skal give interrupt til en vilkårlig processor på et vilkårligt niveau, skal være i stand til at drive 56 forskellige interrupt request ledninger.

4.1. Interrupt systemet.

Enhed A sender interrupt til enhed B ved at skrive i B´s lager. Adressen, hvor A skal skrive, afhænger af A´s enhedsnummer samt prioriteten af det pågældende interrupt.

Enhed B er forsynet med elektronik, der dekoder adressen og detekterer skrivning i bestemte lagerceller. Denne elektronik genererer et interrupt på det ønskede niveau.

En enhed, som skal modtage interrupt fra 8 andre enheder, skal derfor reservere 8*7 bytes i sit lager. Den byte, som skrives ved afgivelsen af interrupt, giver oplysning om interrupt årsagen.

```
enhed 0
enhed 1          Prioritetsniveau 0
..
enhed 7
enhed 0
enhed 1          Prioritetsniveau 1
..
enhed 7
..
..
..
..
enhed 0
enhed 1          Prioritetsniveau 6
..
enhed 7
```

På fig. er vist en skitse af interrupt enheden i en CPU. Bemærk, at MC68000 anvender auto vectoring.

4.2. Begrænsninger i interrupt systemet.

En processor laver en interrupt acknowledge cycle for niveau x og resetter herved den pågældende flip flop. Inden processoren er begyndt at undersøge interrupt årsagen ved at søge i interrupt tabellen, kommer et nyt interrupt på niveau x. Det sidst ankomne interrupt findes sammen med det først ankomne ved søgning i tabellen. Interruptene behandles, lagercellerne resettes, og normal eksekvering genoptages. Det sidst ankomne interrupt på niveau x er behandlet, men flip floppen er ikke resat. Der kommer derfor endnu et interrupt på niveau x, men da ingen interrupt årsag kan findes, returnerer interrupt programmet blot.

En enhed kan kun give et interrupt til en anden enhed på et niveau.

5. CPU.

En processor i SPC/2 består af en CPU samt et antal identiske lagerkort. CPU´en, som søges realiseret på et kort indeholder den fælles del af styringen for det dynamiske lager. Denne styring vil blive beskreve under beskrivelsen af lageret.

CPU´en er baseret på MC68000 og indeholder foruden denne en række mere eller mindre selvstændige enheder:

1. Memory Management Unit, MMU.
2. Interface til fællesbussen.
3. Interrupt enhed.
4. Timer til timer interrupt.
5. Lokale ydre enheder f.eks. USART til diagnose terminal.
6. Styring af dynamisk lager.

5.1. Beskrivelse af MMU.

Formålet med MMU´en er at relokere programmer og data samt at beskytte CPU´ens lager. MMU´ens funktion er at oversætte logiske adresser fra CPU og fællesbus til fysiske adresser, der anvendes ved adressering af lageret. Logiske adresser er de adresser, som kommer fra CPU´en, d.v.s. de adresser, der anvendes i programmerne. Foruden relokeringen kontrollerer MMU´en, at accessen til lageret er tilladt.

Da maskinen skal anvendes i et multiprogrammeret miljø, er det ønskeligt, at relokeringen ikke afhænger af den logiske adresse allene, men også af et bruger/proces nummer. Dette nummer betegnes i det følgende Address Space Number, ASN. ASN er ikke

indført for at udvide adresserummet for MC68000, men derimod for at lette administrationen af lageret.

Ved beskrivelsen af MMU´en er to begreber nødvendige:

En Blok: det minimale antal bytes, som kan relokeres og beskyttes. Ved realiseringen af MMU´en fastsættes bloklængden til mellem 256 og 4k bytes.

Et Segment: et antal logisk og fysisk sammenhængende blokke. At blokkene er både logisk og fysisk sammenhængende betyder, at de relokeres på samme måde. Et segment beskrives ved:

Logisk start adresse.
Offset.
Længde.
Access kode.

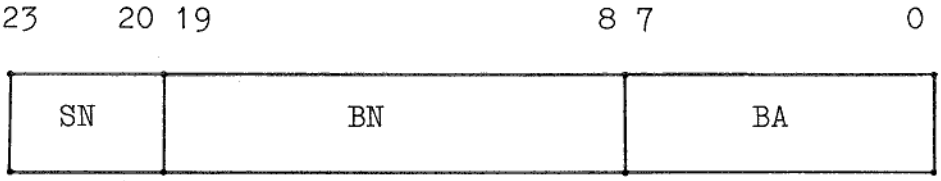
Det er et krav til MMU´en, at en proces defineret ved et ASN skal have adgang til mere end et segment.

MMU´en realiseres ved hjælp af et lager, som vil blive betegnet MMU.

Til opslag i MMU´en anvendes de mest betydende bit af den logiske adresse, hvilket er segment nummeret, og ASN.

Ordet som udlæses af MMU´en indeholder Access kode, længde og offset for segmentet.

5.2. MMU´ens funktion.



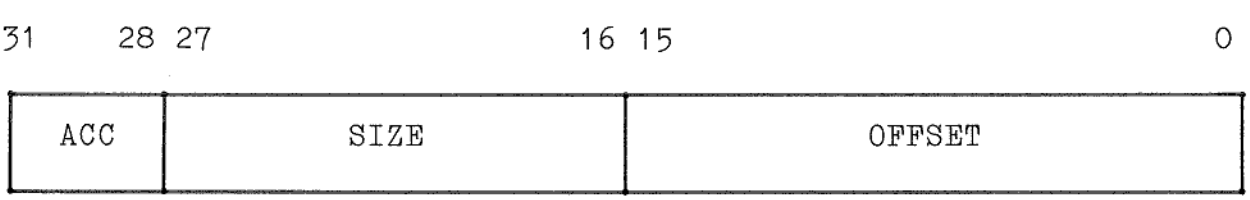
Den logiske adresse A(23:0) opfattes på følgende måde:

A(23:20): SN, Segment nummer. Det logiske adresserum inddeles på dennne måde i 16 segmenter a maksimalt 1 Mb.

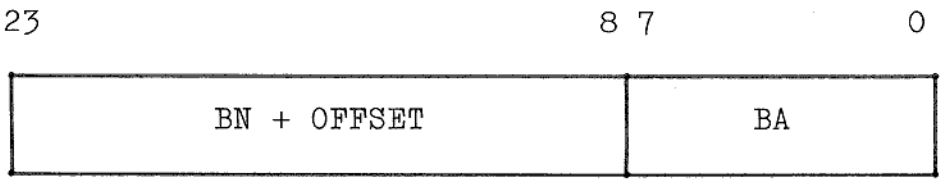
A(19:8): BN, Blok nummer. Et segment består af indtil 4096 blokke.

A(7:0): BA, Blok adresse. en blok består af 256 bytes.

Ved opslag i MMU´en anvendes en 10 bit adresse bestående af A(23:20) og ASN(5:0). Ordet som udlæses af MMU´en er på 32 bit og opfattes på følgende måde:



Den fysiske adresse, som MMU´en genererer, har følgende udseende:



De 16 mest betydende bit af den fysiske adresse genereres ved addition af OFFSET og BN, bloknummer fra den logiske adresse.

BA, blokadressen anvendes uændret som de 8 mindst betydende bit af adressen. Samtidigt med beregningen af den fysiske adresse kontrollerer MMU´en, om accessen er lovlig. Denne kontrol sker ved at sammenligne statussignalerne fra MC68000 med access koden, ACC, og BN med SIZE. Når BN <= SIZE er accessen inden for segmentets område, hvilket betyder, at et segment bestående af en blok har SIZE = 0.

Access koden bestående af 4 bit har følgende betydning:

IO: Access fra fællesbussen er tilladt.
U: User mode er tilladt.
W: Skrivning er tilladt.
X: 0 : Ordet udlæst af MMU anvendes som beskrevet.
1 : Ordet udlæst fra MMU anvendes ved access til fællesbus. Dette beskrives senere.

Som MMU´en er beskrevet hidtil har en process rådighed over indtil 16 segmenter. De logiske adresser inden for et segment oversættes til fysiske adresser og accessen kontrolleres.

Brugerprocesser kan udføre forskellige funktioner ved anvendelse af supervisor call (TRAP). Parametrene til disse funktioner findes generelt i brugerprocessens lager og skal hentes her af supervisoren. Dette kan ske på to måder:

1. Supervisoren kender de relokeringer, der anvendes af brugerprocessen og er derfor i stand til at beregne de fysiske adresser. Supervisoren kan accesse lageret ved anvendelse af fysiske adresser og læser parametrene på denne måde.

2. Supervisoren anvender MMU´en og brugerprocessens ASN og kan direkte accesse de ønskede parametre.

Mulighed 2 er den simpleste løsning, men den kræver at nogle segmenter kun anvendes af supervisoren. Det er besluttet at anvende segment 0 til supervisoren og segment 1 til adressering af diverse enheder.

SEG NR	USER MODE	SUPERVISOR MODE
0	Ulovlig access	Ingen relokering Supervisor
1	Ulovlig access	Ingen relokering Adressering af YE
2	Relokering ASN con SEG	Relokering ASN con SEG
3	Relokering ASN con SEG	Relokering ASN con SEG
.		
.		
15	Relokering ASN con SEG	Relokering ASN con SEG

Når segment nummeret er 2 til 15, foretages relokering ved hjælp af MMU som tidligere beskrevet.

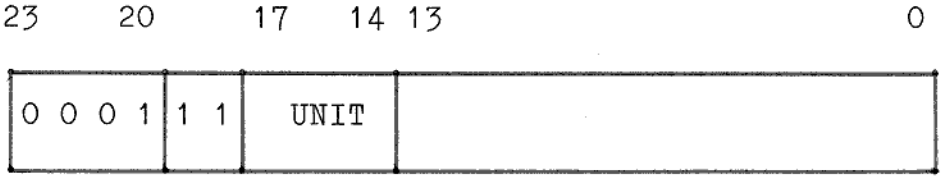
Når CPU er i User Mode og segment 0 eller 1 accesses, genereres interrupt for ulovlig access.

Når CPU er i Supervisor Mode og segment 0 accesses, anvendes ingen relokering.

Når CPU er i Supervisor Mode og segment 1 accesses, laves ingen lageraccess, men adressen anvendes til adressering af diverse ydre enheder herunder MMU'en selv.

5.3. Access til fællesbussen.

Ved access til fællesbussen skal der generelt genereres en 34 bit adresse som beskrevet uder fællesbussen. Da kontrollere kun anvender en del af denne adresse, er det praktisk at lade MC68000 accesse fællesbussen ved anvendelse af en kortere adresse. En del af segment 1 anvendes til dette formål:

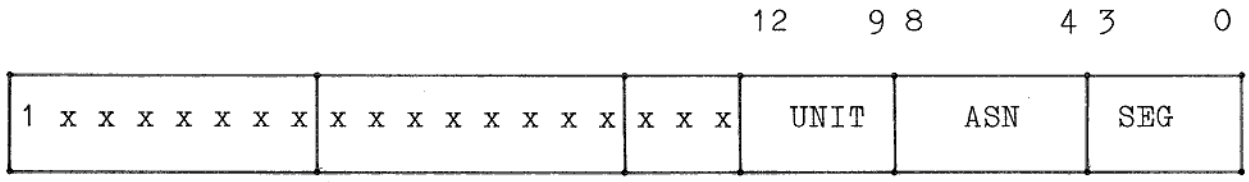


Den sidste del af segment 1 anvendes til at adressere 16 kbytes i 16 forskellige enheder tilknyttes fællesbussen.

Ved dennne access sendes ASN = 0000 ud på fællesbussen.

Denne adresseringsmetode tænkes først og fremmest anvendt til adressering af kontrollere.

Når bit X i access koden fra MMU´en er sat, anvendes det udlæste ord fra MMU´en på en anden måde:



Af det udlæste ord anvendes kun 13 bit. Disse tolkes som UNIT, ASN og SEG og sættes ud på fællesbusseen sammen med BN og BA fra den logiske adresse. På denne måde kan en brugers segment placeres i en anden CPU´s lager dog på bekostning af effekti-

viteten. Denne adresseringsmetode realiseres kun, hvis pladsen på CPU kortet tilladre det.

5.4. Access fra fællesbussen.

Interface til fællesbussen dekoder til stadighed UNIT på fællesbussen. Når CPU´en adresseres fra fællesbussen standser den, når den har afsluttet sin nuværende cycle. Adressen fra fællesbussen indeholdende ASN, SEG m.v. oversættes til en fysisk adresse, som adresserer lageret. Accessen er kun lovlig, når IO i access koden er sat. Hvis accessen er ulovlig, skal datatransporten standses. Den simpleste løsning er blot at generere et interrupt i CPU´en, som så selv må forsøge at finde ud af hvem, der har forsøgt at foretage en ulovlig access. Denne undersøgelse kan blive ret kompliceret, hvorfor den ulovlige access meddeles til den enhed, som har overtrådt lagerbeskyttelsen. Det er da denne enheds opgave at sende interrupt til CPU´en, hvorefter CPU´en kan læse de nødvendige data i enhedens lager.

Ved input output til en process kan kontrolleren anvende processens ASN, hvorved et helt segment er åbent for input output. Hvis man kun ønsker at åbne een mindre del af segmentet for input output, kan kontrolleren anvende et andet ASN, som ved hjælp af MMU´en afbildes ind i processens segment.

6. Lager.

Lageret har en ordlængde på 16 bit, men tillader læsning og skrivning af en byte. Lageret udføres med fejlkorrektion, hvorved der skal tilføjes 6 bit ekstra til lagerets ordlængde. Ved denne fremgangsmåde kan man korrigere alle enkeltfejl og detektere alle dobbeltfejl. Enheden, som kontrollerer data og generer check bit, kaldes EDC.

Hvis EDC detekterer enkeltfejl, korrigeres data, og der genereres et interrupt til CPU'en.

Hvis der detekteres enkeltfejl i lageret ved access fra fællesbussen, korrigeres data, og der genereres et interrupt til CPU'en. Fejlen bør også meddeles til kontrolleren.

Hvis der detekteres dobbeltfejl, gives Bus Error Exeption til 68000. Denne exception behandles som et interrupt dog med den modifikation, at 68000 går i HALT, hvis der optræder en ny Bus Error under skrivning af status og læsning af ny status for interrupt rutinen.

Ved fejl skal EDC indeholde oplysninger, som lokaliserer fejlen til en enkelt lagerkreds. Disse oplysninger er tilgængelige for programmellet.

Læsning.

Ved læsning i lageret kontrolleres de udlæste data. Ved fejl korrigeres data, og de korrigerede data skrives i lageret igen. Der genereres interrupt.

Skrivning af ord.

EDC genererer checkbit, som skrives i lageret sammen med data.

Skrivning af byte.

Ordet i lageret udlæses og kontrolleres. Hvis der optræder fejl, korrigeres data, og den byte, der skal skrives i lageret, indsættes i det korrigerede ord. Dernæst generes checkbit, og ordet skrives i lageret. Ved fejl genereres interrupt til CPU'en.

Lageret pakkes på printkort med minimum 1/2 Mbytes pr. kort.

Lagerstyringen udføres med synkrone sekvensmaskiner i lighed med SPC/1. Klokfrekvensen for disse er 16-20 Mhz.

7. Input/Output.

Input Output varetages af intelligente kontrollere, IOP, som kommunikerer med processorerne over fællesbussen. De intelligente kontrollere udføres med 8085 og periferikredse kendt fra SPC/1. Der findes 3 typer input/output:

1. Direkte input.
2. Blok input.
3. Blok output.

Datatransporten i en input output transaktion består af to faser:

A. Transport mellem processor og kontroller.
B. Transport mellem kontroller og ydre enhed.

Disse faser kan følge efter hinanden eller forløbe parallelt afhængigt af buffer mekanismen i kontrolleren. Begge faser styres af kontrolleren, d.v.s. i fase A overfører kontrolleren data mellem processorens hovedlager og sit eget lager, idet adresseringen af processorens lager sker gennem dennes MMU. For at fase A kan forløbe korrekt, skal MMU´en altså være initialiseret korrekt, hvilket udføres af processoren. Under fase A er dele af MMU´en beslaglagt af beskrivelsen af data transporten.

En bloktransport sker på følgende måde:

OUTPUT

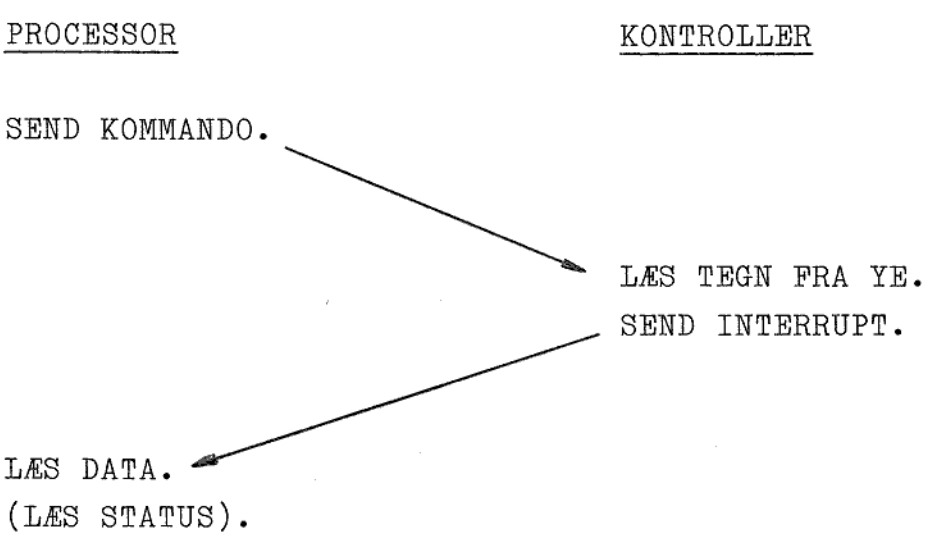
PROCESSOR	KONTROLLER
INITIALISER MMU.	
SEND PARAMETRE.	
SEND KOMMANDO. →	
	LÆS DATA I PROC LAGER.
	SKRIV DATA TIL YE.
	SEND INTERRUPT TIL PROC. →
LÆS STATUS.	
FRIGØR MMU.	
FRIGØR YE.	

INPUT

PROCESSOR	KONTROLLER
INITIALISER MMU.	
SEND PARAMETRE.	
SEND KOMMANDO.	
	LÆS DATA FRA YE.
	SKRIV DATA I PROC LAGER.
	SEND INTERRUPT.
LÆS STATUS.	
FRIGØR MMU.	
FRIGØR YE.	

Direkte input er karakteriseret ved, at et enkelt tegn overføres fra den ydre enhed til processoren.

DIREKTE INPUT



I dette tilfælde styrer processoren transporten fra kontroller til processor.

7.1. I/O processorer.

En I/O processor indeholder et antal kanaler. Ved en kanal forstås de registre, lagerceller, der er nødvendige for at understøtte en I/O transaktion. En I/O kanal kan være i en af følgende tilstande:

1. LEDIG.

Kanalen indeholder ingen information.

2. OPTAGET. PASSIV.

Kanalen indeholder information, men deltager ikke i en I/O transaktion. Kanalen er i denne tilstand, mens CPU´en overfører parametre til kanalen, samt når transporten er afsluttet, men inden CPU´en har frigjort kanalen.

3. OPTAGET. AKTIV.
Kanalen indeholder information og deltager i en I/O transaktion.

En kanal, som deltager i en disktransport kan f.eks. indeholde følgende information:

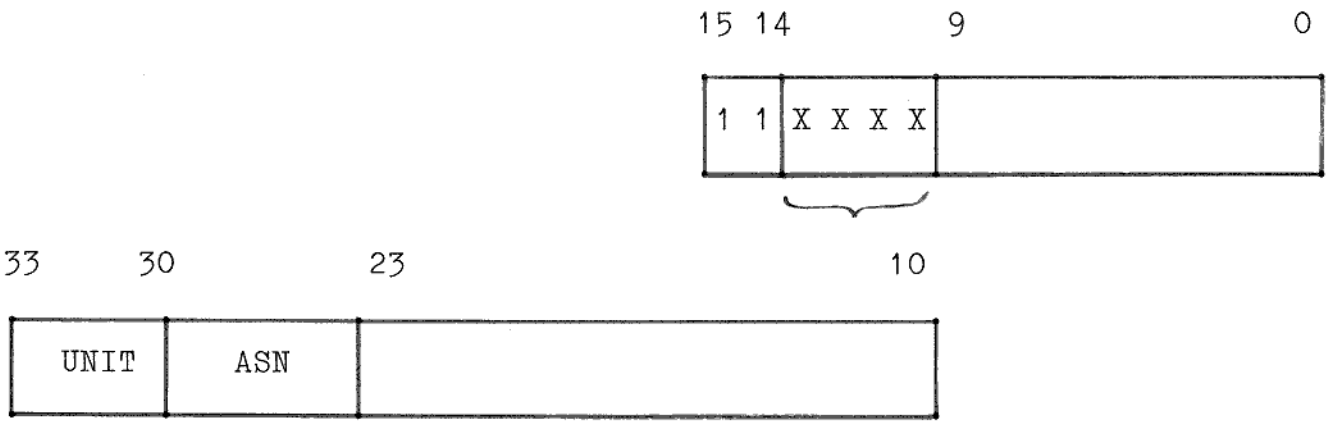
UNIT:	Enhed, hvortil transporten foregår.
ASN:	Address space number for transporten.
LAGERADRESSE:	Lageradresse i CPU.
KONTROLLER:	Adresse på disk kontroller.
DISK:	Adresse på disk drev.
SEKTOR:	Sektornummer.
ANTAL:	Antal sektorer i overførsel.
KOMMANDO:	Læs/skriv m.v.
STATUS:	Status for overførsel.

Antallet af kanaler i en kontroller kan godt være større end antallet af kanaler, der kan være aktive samtidigt.

7.2. SIOK.

SIOK er en seriel input/output kontoller. Den er karakteriseret ved, at den kan have adskellige langsomme transporter forløbende samtidigt. De enkelte transporters hastighed er forholdsvis lille sammenlignet med busbåndbredden. En SIOK kan f.eks. drive 8 transporter samtidigt. Når en SIOK skal adressere en CPU´s lager, skal den generere en 34 bit adresse.

Dette gøres ved at ekspandere 8085´s adresserum:



De sidste 16 kbytes i 8085 logiske adresserum ekspanderes ved hjælp af en memory mapper, idet det inddeles i 16 blokke a 1 kbytes. Adressen, hvor en blok skal læses eller skrives, findes ved opslag i memory mapperen, som er et lager på 16 ord a 24 bit. Data fra lageret sættes foran de 10 mindst betydende bit af 8085´s adresse, hvorved den logiske adresse til adressering af fællesbussen genereres.

7.3. VIOK.

En VIOK er en diskkontroller. Den adskiller sig fra en SIOK ved, at den kun kører nogle få, eventuelt kun en transport ad gangen, og transporten sker med høj hastighed.

I det følgende skitseres en VIOK, som driver en SASI bus.

VIOK består af 8085 med lager samt en programstyret interface til SASI bussen i lighed med adapteren, som anvendes i SPC/1.

Kommandoer overføres til SASI bussen ved hjælp af denne interface. Foruden den programstyrede interface findes en interface, som slutter SASI bussen til fællesbussen. Denne inter-

face indeholder registre til UNIT, ASN, SEG, en 20 bit tæller til lageradressen og et 16 bit data register. Denne interface initialiseres under program kontrol. Når dette er sket, kan den selv udføre handshake med SASI bussen og fællesbussen under datatransporten. Ved læsning fra disken udfører denne interface handshake med SASI bussen for hver byte, som læses fra bussen. Når interface har læst og pakket 2 bytes i data registeret, skrives disse i CPU´ens lager over fællesbussen, idet interfacens registre anvendes ved adresseringen. Ved hver overførsel over fællesbussen tælles adresseregisteret to op.

En sådan interface kan udføre en transport ad gangen.

APPENDIX 1.

Beregning af CPU cycle.

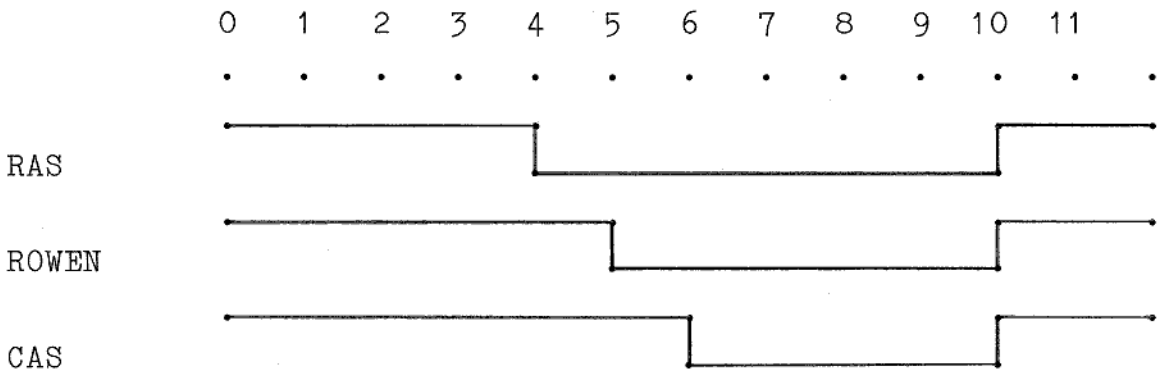
Langt den hyppigste CPU cycle er en lager læsning, hvorfor kun denne cycle vil blive betragtet i det følgende.

I en CPU cycle sker der følgende ting:

1. CPU leverer adressen og styresignaler.
2. Adressen oversættes i MMU´en.
3. Læsning i lageret startes.
4. Data fra lager indlæses i EDC.
5. EDC sender DTACK, data OK, til CPU.
6. CPU læser data.

For at opnå den hurtigste CPU cycle skal man normalt anvende den højst mulige klokfrekvens til CPU´en og dernæst indskyde det antal WAIT states, der er nødvendige af hensyn til lageret. I det følgende regnes derfor med, at 68000 kører på en 8 Mhz klok. Lagerstyringen antages udført synkront på en 20 Mhz klok.

Før lageret kan startes skal adressen oversættes i MMU´en, hvilket tager 150 ns svarende til 3 klok. Timingen af lageret har følgende udseende:



0: Adressen og styresignaler er klar.
0-1: Worst Case synkroniserings delay.
1-4: Adressen oversættes i MMU.
9: Data samples i EDC.
10: DTACK sendes til 68000.

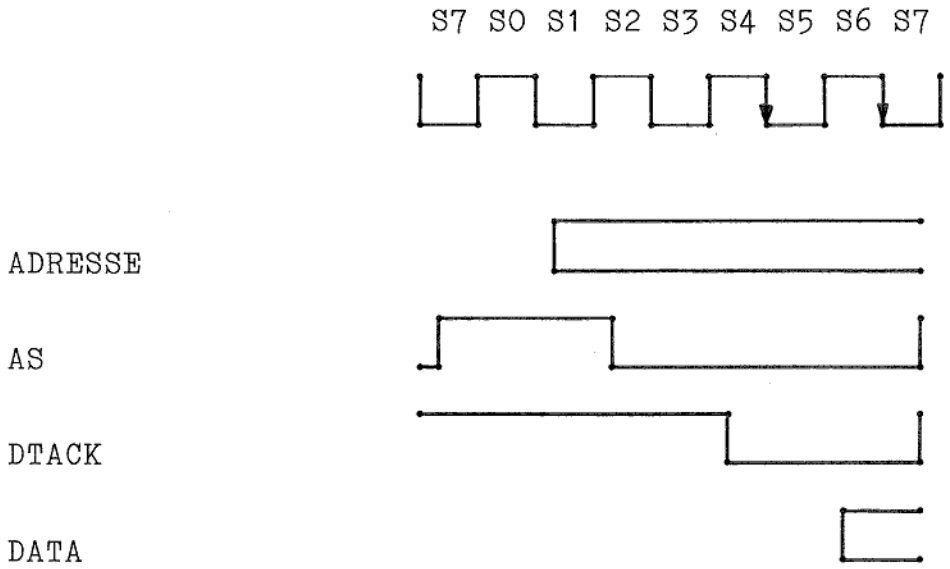
Access tid fra adresse er klar, til data er kontrolleret:

500 ns.

Cycle tid for lager: 500 ns.

Tiderne inkluderer oversættelse af adresse i MMU og kontrol af data i EDC.

Basis timing for 8 Mhz 68000.



Adresse er klar 70 ns efter start af S1.
AS low 60 ns efter start af S2.
DTACK samples på start af S5.
DATA samples i start af S7.

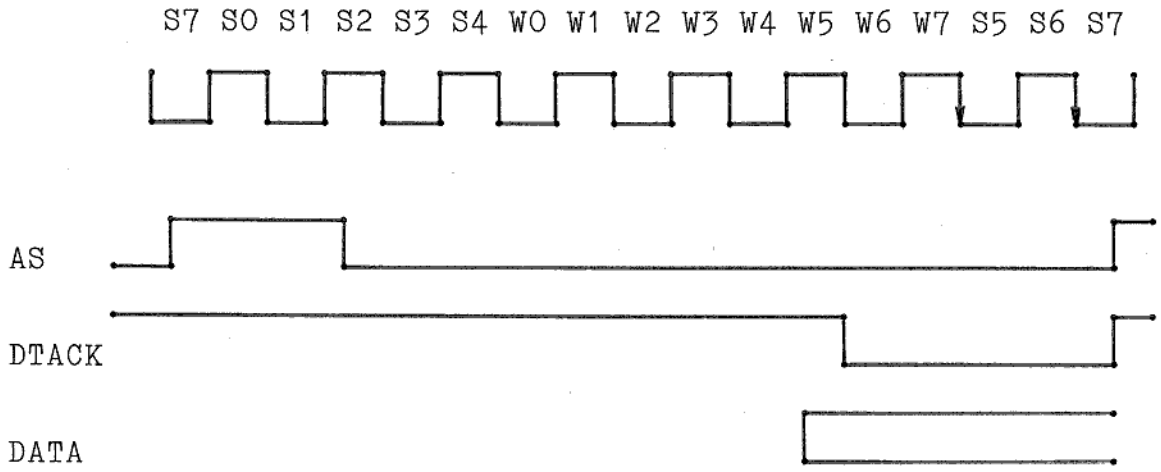
Lageret kan først startes efter AS low. Lageret kan derfor først sende DTACK 60+500 ns = 560 ns efter S2.

Hvis der indskydes 3 WAIT klok cycles, sampler 68000 DTACK 9*62.5 ns = 562.5 ns efter S2. Dette giver kun en set up tid på 2.5 ns for DTACK, hvilket ikke er tilstrækkeligt.

Der indskydes 4 WAIT klok cycles.

Cycle tiden for 68000 bliver herved 1 us.

Resulterende timing.



DTACK samples ved starten af S5.

Data samples ved starten af S7.

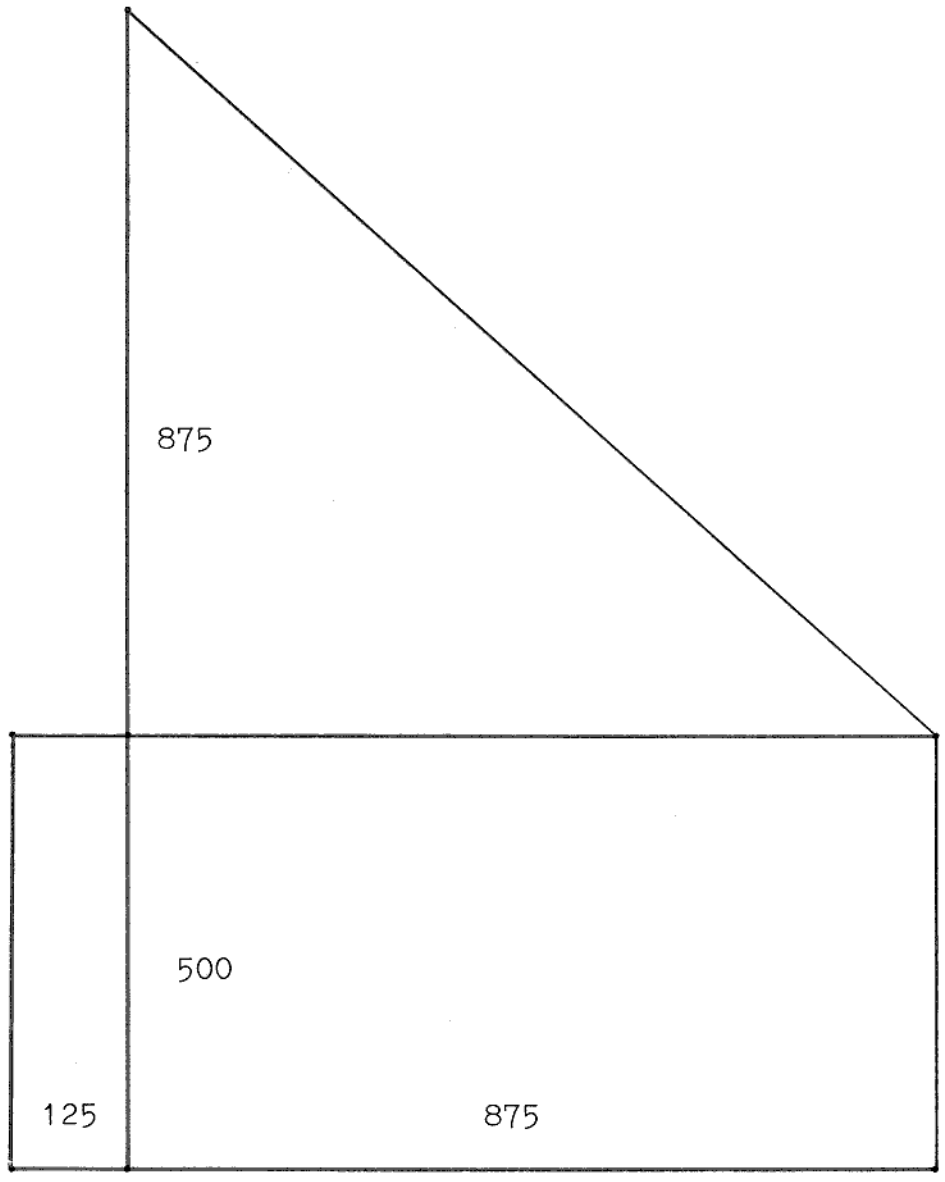
CPU cycle: 1 us.

Da lagerstyringen kører asynkront med 68000, vil en læsning undertiden kun tage 875 ns.

Bemærk, at CPU´ens hastighed er halveret på grund af lageret.

Access til lager fra fællesbus.

Når accessen fra fællesbussen kommer i S0 eller S1, udføres den straks, ellers ventes til CPU har afsluttet sin cycle.



Maksimal access tid fra fællesbus: 1375 ns.
Middel access tid fra fællesbus: 882 ns.