NTNU
Kunnskap for en bedre verden

Institiutt for informatikk og e-læring, NTNU

CPUens deler og virkemåte

Geir Ove Rosvold
4. januar 2016

CPUens deler og virkemåte

Resymé: Denne leksjonen beskriver CPUens deler og virkemåte. Vi ser at en instruksjon utføres ved at den brytes ned til mindre og enkle handlinger; såkalte mikrooperasjoner. Vi ser hvordan disse mikrooperasjonene kan se ut på vår hypotetiske maskin.

Innhold

CPUens oppbygging

Så langt i faget har vi stort sett kikket på de forskjellige delene i datamaskinen, og diskutert hvordan de samvirker. Vi skal gå over til å se på det som er selve hjernen i datamaskinen; nemlig CPUen eller prosessoren.

CPU er en elektronisk krets som er i stand til utføre instruksjoner. Disse instruksjonene ligger på et lavt abstraksjonsnivå og er grunnleggende og enkle av natur. Et eksempel er ADD-instruksjonen på vår hypotetiske maskin. Nå skal vi se nærmere på hvordan CPUen er bygget opp, og hvordan den virker.

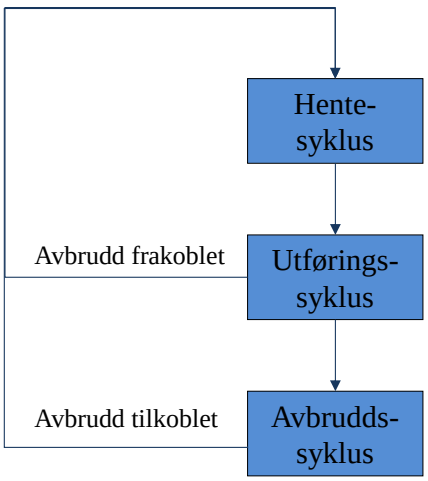
Instruksjonssyklusen

La oss begynne med en kort repetisjon av instruksjonssyklusen. I leksjonen om data-maskinens grunnleggende virkemåte så vi hvordan CPUen veksler mellom hente- og utføringssyklusen.

I hentesyklusen hentes en ny instruksjon fra minnet. Instruksjonen legges i IR. I utføringssyklusen utfører CPU denne instruksjonen.

Senere innførte vi også avbruddsyklusen. Når du fortsetter å lese denne leksjonen, er det viktig å huske at CPU normalt veksler mellom hente- og utføringssyklusen.

I avbruddssyklusen skjer det normalt ingenting. Unntaket er når det har kommer et avbrudd. Men der er tross alt relativ sjelden.



Figur 1. Instruksjonssyklusen består av tre delsykluser. Den vanlige instruksjonsutføringen består i at CPU veksler mellom hente- og utføringssyklys. Ved avbrudd vil avbruddssyklusen sørge for at rett avbruddsrutine starter. I kritiske deler av et program kan avbruddsmekanismen frakobles, men dette er sjelden.

De viktigste delene til CPU

De viktigste delene av CPUen er:

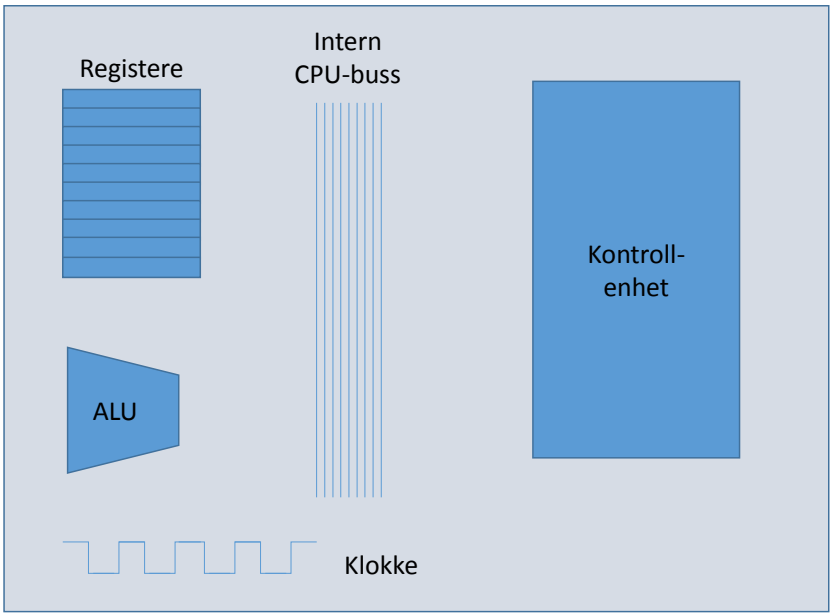
- Registerblokken (det vil si samlingen av alle registre på en CPU)
- ALU (Aritmetisk/logisk enhet)

- Kontrollenheten
- Klokkekretsen
- Intern CPU-buss

De ulike delene kommuniserer over den interne CPU-bussen. Instruksjonene utføres i steg som startes av klokkepulser som kommer med konstant frekvens. Antall steg en instruksjon trenger for å bli ferdig varierer - noen instruksjoner trenger mange, andre trenger få .

Utføringen skjer under kontroll av kontrollenheten som setter opp de riktige styresignal til enhver tid.

Sett fra programmererens side er nok registerblokken den viktigste delen, eller i hvert fall den delen man oftest må forholde seg til. Derfor begynner vi med å se på den. Senere skal vi se på de andre delene også



Figur 2 CPUens viktigste deler. Registrene, ALUen, klokkekretsen, den interne CPU-bussen og kontrollenheten er de viktigste delene av prosessoren.

Registerblokken

Ut fra det som er sagt så langt i kurset er det vel lett å se at registrene spiller en viktig rolle for CPUens virkemåte. Hvilke registre som finnes, og hvor mange de er, vil alltid være en viktig del av CPU-designet.

Det finnes mange ulike typer CPUer. De mest kjente produsentene er kanskje Intel, AMD, IBM, Motorola, Texas Instruments og Nvidia – med Intel som den aller største. Produsentene har laget et utall forskjellige CPU-typer der både antall registre og måten de er organisert på varierer. En del CPUer har relativt få registre, f.eks noen titalls, mens andre CPUer kan ha flere hundre av dem.

Samlingen av alle registre på en CPU kalles *registerblokken*. Når vi snakker om register-blokken mener vi altså alle registrene på prosessoren.

Det finnes mange forskjellige registre på en CPU. En del registre kan aksesseres av bruker-programmene (det vil si at programmereren har tilgang til dem, både via enkelte høynivå-språk, assemblyprogrammering og via debuggere). Andre registre er usynlig for bruker-program og benyttes bare av opeativsystemet eller internt av CPUen.

Antall registre som finnes av hver type varierer fra CPU-type til CPU-type. Bredden på registrene, bruken av generelle registre og så videre varierer også mellom CPU-typene.

Registre som er tilgjengelig for brukerprogrammene

De viktigste typene av bruker-tilgjengelige registre er:

- **Generelle registre**. Disse brukes til data som CPU trenger under utføring.
- **Flagg-registeret**. Dette er et register som bare kan leses, og som angir status for CPUen.
- **Adresseregistre**. Adresseregistre inneholder adresser til minnet. Det er altså registre som peker til ulike minnelokasjoner. Det finnes flere sorter av slike registre: Segmentregistre peker til forskjellige deler av minnet. Indeksregistre med eller uten automatisk opptelling brukes til indeksering i tabeller og arrayer.

Registre som ikke er tilgjengelig for brukerprogrammene

Dette er registre som brukeren ikke kan aksessere, men som brukes av CPU. De viktigste registrene av denne typen er:

- Programteller
- Instruksjonsregister
- Registre som brukes av operativsystemet og som kun kan aksesseres av priviligerte instruksjoner
- Interne registre som brukes til bufring under utførelse av instruksjoner

Kontrollenheten

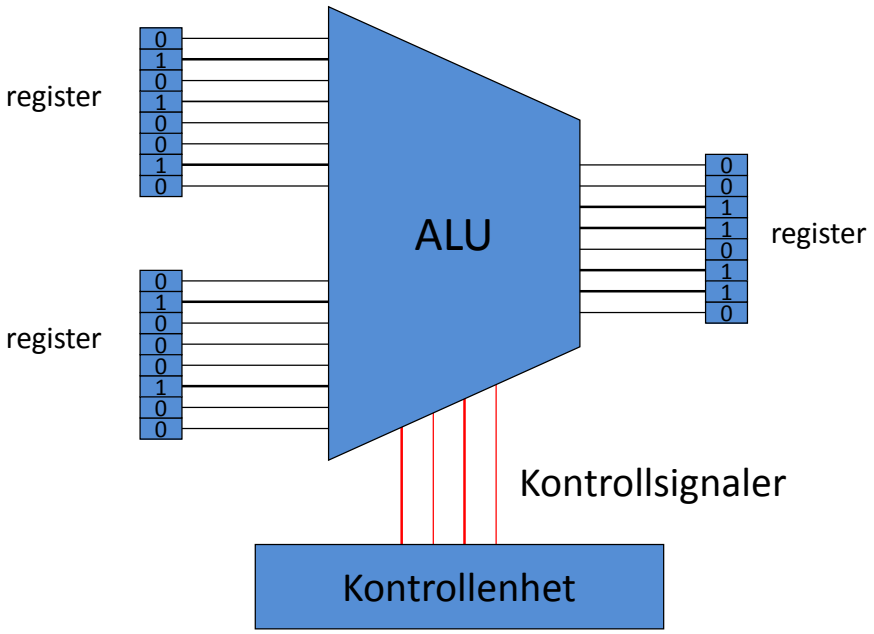
Aktivitetene til CPU styres av en kontrollenhet som sender kontrollsignaler til de andre enhetene. Disse kontrollsignalene er binære signaler; de er enten av eller på. Eksempler på kontrollsignaler er:

- Kontrollsignaler som bestemmer hvilken ALU-operasjon som skal foretas.
- Kontrollsignaler som åpner/lukker for dataflyt mellom to registre.

Vi skal se nærmere på kontrollenheten senere i leksjonen.

ALU

ALU er en komponent som kan utføre et antall aritmetiske og logiske operasjoner på binære data. ALUen har to innganger og en utgang. Både innganger og utganger er registre. ALU kombinerer bitmønstrene på inngangen og det resulterende bitmønsteret presenteres på utgangen. De to inngangene kan kombineres på mange måter, for eksempel kan de adderes, subtraheres, multipliseres, divideres (aritmetiske operasjoner), eller de kan sammenlignes på ulike måter (logiske operasjoner). Det er kontrollsignaler som bestemmes hvilken operasjon som skal utføres. Figur 3 viser oppbyggingen til en ALU.



Figur 3 ALU. ALU er en komponent som kombinerer bitmønstrene på de to inngangene og legger resultatet på utgangen. Kontrollsignalene er tegnet som røde linjer i figuren. Med fire kontrollsignaler som hver er enten av eller på kan vi velge mellom $2^4 = 16$ ulike operasjoner.

Klokkekretsen

Alle handlinger som skjer i en CPU startes av en klokkepuls. Klokkepulser er elektroniske pulser som kommer fra en såkalt klokkekrets. Klokkekretsen er bygget rundt et kvartskrystall, og gir pulser med faste mellomrom (konstant frekvens).

Frekvensen til disse klokkepulsene er i området 300-4000 Mhz for de fleste prosessorer[1]. Det betyr at det aktiviseres en ny handling mellom 300 millioner og 4 milliarder ganger i sekundet.

I reklamesammenheng var man for noen år siden svært opptatt av klokkefrekvensen når man skulle vurdere prosessorer. Man tok utgangspunkt i at klokkefrekvensen hadde stor betydning for ytelsen. Dette var i stor grad en reklamestyrt oppfatning. I realiteten er det viktigste hvor mye arbeid man får utført i løpet hvert klokkeslag. På slutten av 90-tallet var det et kappløp om å produsere den første GHz-prosessoren (AMD vant dette kappløpet med sin Athlon-prosessor i mars 2000). Senere har man heldigvis gått bort fra dette ensidige fokuset på klokkefrekvens. De siste årene har økt ytelse i størst grad kommet av at man får utført mer arbeid pr klokkesyklus, f.eks ved å utstyre prosessoren med flere kjerner. Men mer om dette senere i kurset.

[1] Man skulle kanskje tro at stormaskiner og supermaskiner hadde en mye høyere klokkefrekvens enn mikromaskiner. Dette er imidlertid ikke tilfellet. De hurtigste mikroprosessorene, f.eks. fra Intel eller AMD, er minst like hurtig som de fleste stormaskiners CPU. Forskjellene ligger først og fremst i oppbyggingen, blant annet hvor mye som foregår i parallell, og ikke minst hvor mye I/O prosessorene tar hånd om (en stormaskins prosessor avlastes med avanserte I/O-moduler og er i liten grad involvert i I/O).

Intern CPU-buss

Når data skal sendes mellom de ulike deler av CPUen benyttes en intern CPU-buss. Dette er en svært hurtig parallell buss. Siden den er en integrert del av CPUen, og i tillegg overfører data over svært korte avstander, benytter bussen CPUens klokkefrekvens. Alle de deler av CPUen som har behov for å utveksle data er tilknyttet den interne CPU-bussen. Eksempler på slike deler er registrene og ALUens innganger og utganger.

Mikrooperasjoner

Nå skal vi se nærmere på hvordan CPUen virker. Vi har allerede nevnt at aktivitetene til CPU styres av en kontrollenhet som sender kontrollsignaler til de andre enhetene. Disse kontroll-signalene er binære signaler; hvert av dem er enten av eller på.

Klokkekretsen gir pulser med konstant frekvens. Kontrollenheten setter opp nye kontroll-signaler hver gang det kommer en ny puls fra klokkekretsen. Disse kontrollsignalene aktiviserer handlinger i CPUen.

De aktivitetene en CPU utfører i løpet av en slik klokkepuls kalles en *mikrooperasjon*. Mikro-operasjoner er den fineste oppdelingen vi bruker på CPUens arbeid - ihvertfall i dette kurset.

Å utføre en instruksjon består i å utføre en sekvens av slike mikrooperasjoner.

Typer av mikrooperasjoner

Utføringen av en instruksjon baserer seg altså på at instruksjonen brytes ned til en sekvens av handlinger som er så enkle at CPU kan utføre hver enkelt handling i løpet av en klokkesyklus. Dette må derfor bli svært enkle handlinger. Alle mikrooperasjoner faller innenfor en av følgende grupper:

1. Flytte data mellom to registre
2. Utføre aritmetiske eller logiske operasjoner med ALU. Input data til ALU ligger i registre, og ALU legger resultatet i et register.
3. Flytte data mellom registre og minne eller IO-registre (I/O-porter) via systembussen.

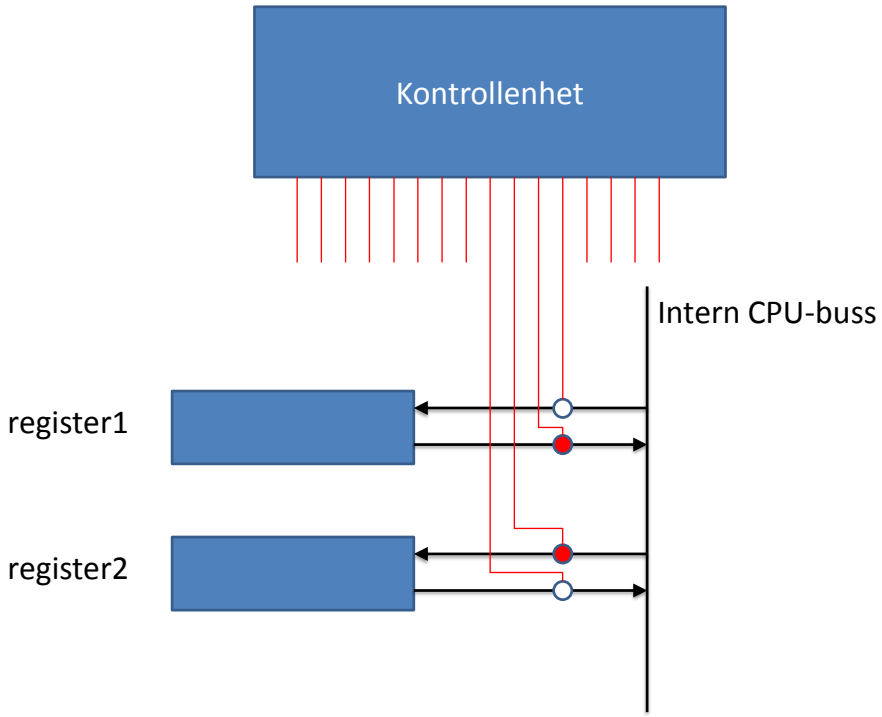
Alle handlinger utføres ved at kontrollenheten setter styresignal på de enkelte enheter i CPUen.

La oss se litt mer i detalj hvordan hver av disse tre typene mikrooperasjoner faktisk kan utføres på en CPU.

Flytte data mellom registre

Når to registre skal utveksle data benyttes de den interne CPU-bussen til dette. Alle registrene er nemlig koblet til denne bussen.

Det er kontrollenheten som styrer overføringen ved å sette de rette kontrollsignaler. Alle registrene har brytere som åpner og lukker for dataflyt mot bussen. Hver bryter styres av et binært kontrollsignal. For å åpne for dataflyt mellom to registre (la oss si fra *register1* og til *register2*) åpner bryteren **ut fra** register1 og bryteren **inn til** register2. Alle andre brytere sperrer for dataflyt. Dette er vist i Figur 4.



Figur 4 Bitmønsteret i et register skal kopieres til et annet. Dette skjer ved at det åpnes for dataflyt mellom de registrene. Det er brytere som åpner og lukker for dataflyt. Bryterne er tegnet som små sirkler i figuren. De to bryterne som åpner for dataflyt er er fyllt med rødt. Alle andre brytere sperrer for dataflyt (hvit fyllfarge). De røde linjene er kontroll-linjer fra kontrollenheten. Hver kontroll-linje overfører et binært kontroll-signal. En kontrollenhet har mange kontroll-linjer, men figuren viser bare bruken av fire av dem.

Bruke ALU

ALUen utfører aritmetiske og logiske operasjoner. En ALU har to innganger og en utgang. Operasjonene kombinerer to bitmønstre som ligger på hver sin inngang, og produserer et resultat på utgangen. Begge innganger samt utgangen er registre.

ALUens funksjon er illustrert i Figur 3. Der ser vi det er kontrollsignaler fra kontrollenheten som bestemmer hvilken aritmetisk eller logisk operasjon som skal utføres. I figuren styres ALUen av fire binære kontrollinjer. Med fire binære signaler kan vi skille mellom 16 ulike operasjoner. ALUen i figuren kan altså utføre 16 ulike operasjon.

Kommunisere med systembussen

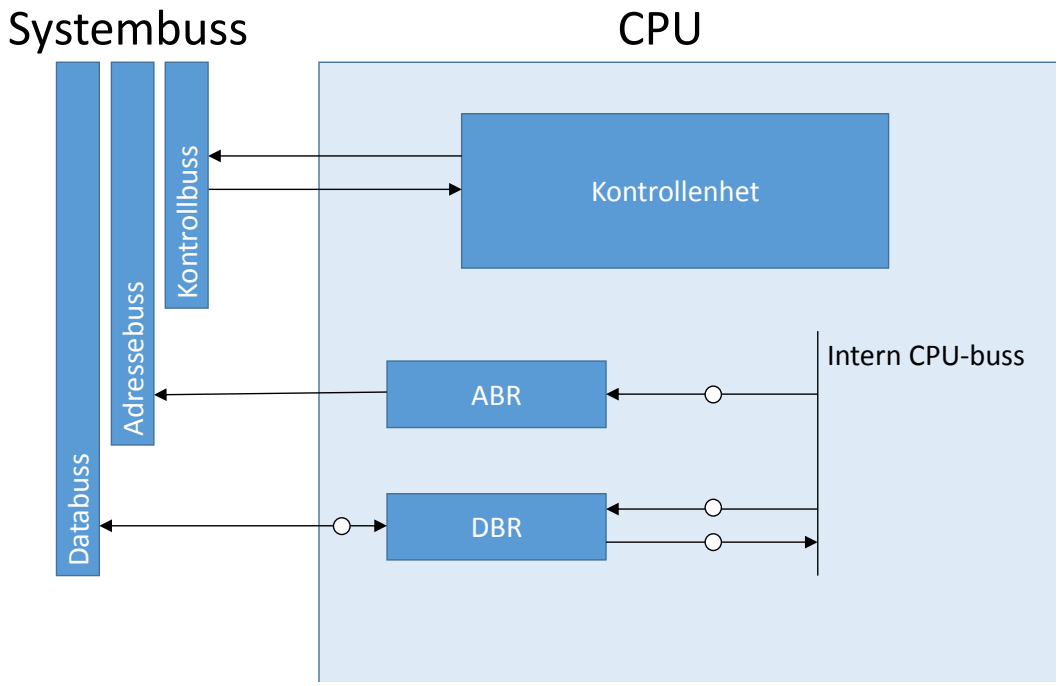
For at CPUen skal kunne gjøre noe fornuftig må den kunne kommunisere med omverden. Denne kommunikasjonen skjer over systembussen. Det vil si bussen som knytter CPU til resten av datamaskinen.

Hvordan skal så CPU lese eller skrive bitmønstrene som ligger på denne bussen? Jo, den gjør det ved hjelp av to spesialregistre som bare har som oppgave å gjøre linjene på systembussen tilgjengelig for CPU:

- **ABR-register** (AdresseBuss Register) – Et register som er tilknyttet adressebussen på systembussen. Registeret inneholder adressen til en minnelokasjon. CPU setter adresselinjene til bussen ved å skrive et bitmønster på ABR-registeret.

- **DBR-register** (DataBuss Register) - Et register som er tilknyttet databussen på systembussen. Dersom CPU vil **skrive** noe til databussen, så skjer det ved at CPU legger bitmønsteret på DBR. Hvis CPU skal **lese** fra minnet, vil minnet legge et bitmønster på databussen, og CPU kan lese det ved å lese DBR-registeret.

Dette er vist i Figur 5.



Figur 5. Kommunikasjon mot systembussen. CPU kommuniserer med systembussen via spesialregistrene ABR (Adressebussregister) og DBR (Databussregister). Hver linje på systembussen er knyttet til den korrensponderende bitposisjonen i ABR og DBR. Kontrollbussen er knyttet til kontrollenheten. Det er kontrollenheten som setter eller leser linjene på kontrollbussen.

Hypotetisk maskin

I tidligere leksjoner har vi diskutert en hypotetisk maskin som var svært enkel, men som viste de viktigste prinsippene. Nå skal vi ta utgangspunkt i denne maskinen, og beskrive den grunnleggende virkemåten til CPUen på den.

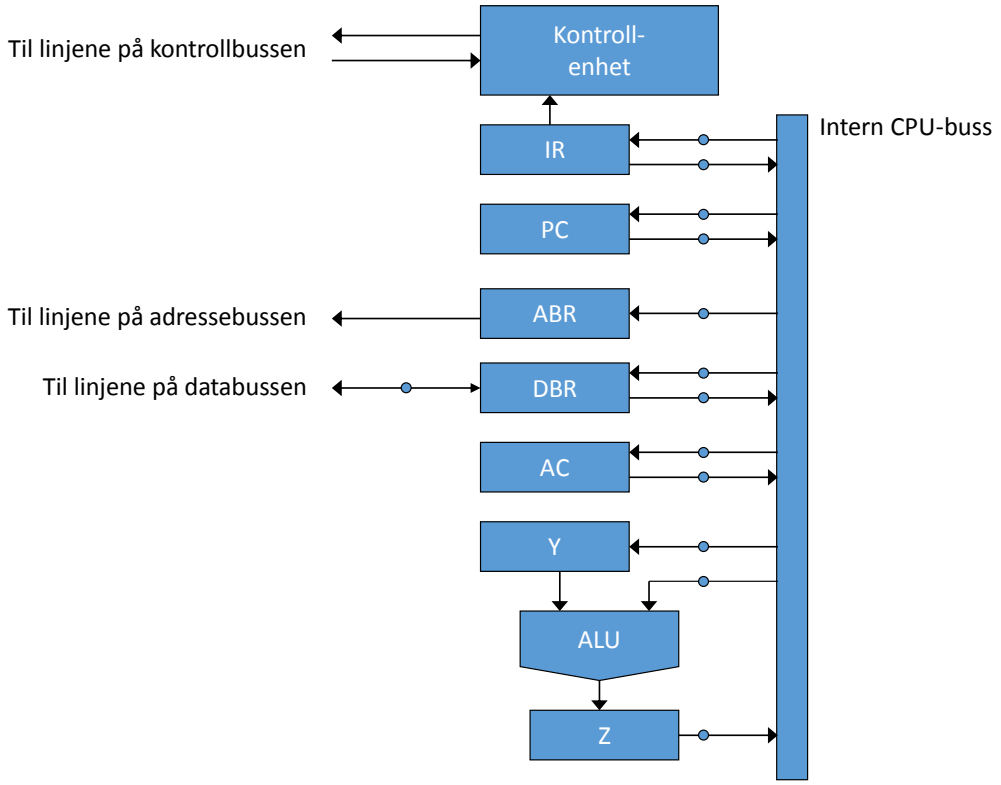
CPUen består av de delene som er nevnt ovenfor. I tillegg finner man blant annet:

- **IR-register** (Instruction Register. Instruksjonsregisteret på norsk) - Et register som inneholder bitmønsteret til en instruksjon som skal utføres
- **PC-register** (Program Counter. Programteller på norsk) - Et register som inneholder adressen til neste instruksjon

I tillegg er det ALU, klokke og en intern CPU-buss. Klokken gir klokkesignal med konstant frekvens og ALUen kan utføre logiske og aritmetiske operasjoner på registerinnhold. De ulike delene av prosessoren kommuniserer over en intern CPU-buss.

Figur 6 viser oppbyggingen. To nye registre, Y og Z, trenger en forklaring. Disse to registrene brukes til bufring eller mellomlagring av data til og fra ALU. De trengs fordi bare ett signal

kan overføres over CPU-bussen i gangen, mens ALU krever to inngangssignal og et utsignal samtidig. (jmf Figur 3).



Figur 6. Bruk av intern CPU-buss. Delene til CPUen kobles sammen via en intern CPU-buss. Ringene er endepunkt for kontrollsignalene som åpner og lukker for dataflyt. Legg merke til at ingen kontroll-linjer er tegnet inn i figuren.

Hentesyklus

Vi skal se hvordan hentesyklusen kan utføres på denne CPUen.

Utgangspunktet er altså at vi akkurat skal begynne på en ny instruksjon som ligger i minnet. PC inneholder adressen til instruksjonen. I hentesyklusen skal instruksjonen hentes fra minnet og legges i IR.

Da må CPU utføre følgende mikrooperasjoner:

1. Innholdet av PC kopieres til ABR. På den måten legges adressen ut på adressebussen. Kontrollenheten setter også de rette linjene på kontrollbussen, slik at minnet får beskjed om at det skal leses og at adressen på adressebussen er gyldig (se leksjon om busser). Dermed kan minnet hente innholdet av denne minnelokasjonen og legge innholdet ut på databussen.
2. Når minnet har lagt innholdet av minnelokasjonen ut på databussen havner den i DBR siden hver enkte bit i DBR er tilknyttet den korrensponderende biten på databussen.
3. DBR kopieres til IR.
4. PC inkrementeres med en, slik at vi henter neste instruksjon neste gang.

Aktivitetene styres av en klokke med fast frekvens. Hver klokkepuls definererer en tidsenhet, og hver mikrooperasjon utføres i løpet av en slik tidsenhet. Hentesyklusen til vår CPU krever altså 4 mikrooperasjoner.

Utføringssyklusen

Utføringssyklusen vil selvfølgelig variere fra instruksjon til instruksjon. Som et eksempel kan vi bruke ADD-instruksjonen til den hypotetiske maskinen. ADD-instruksjonen ble introdusert i leksjonen om grunnleggende virkemåte.

ADD-instruksjonen adderer to tall. Det ene tallet ligger det generelle registeret som kalles akkumulator (forkorter til AC). Vi tenker oss instruksjonen ADD 941. Her er 941 en minne-lokasjon. Instruksjonen legger altså sammen to tall der det ene tallet ligger i registeret AC og det andre tallet ligger i minnelokasjon 941. Resultatet av addisjonen skal legges i AC.

Som vi så i leksjonen om grunnleggende virkemåte, er instruksjonene (bitmønsteret i IR-registeret) delt opp i to: en del angir opkoden, og en del angir en adresse i minnet.

Vi kan tenke oss følgende mikrooperasjoner for utføringssyklusen til denne instruksjonen:

1. Det åpnes for dataflyt mellom adresse-feltet i IR og ABR, slik at adressen til data havner på adressebussen. Kontrollenheten setter også de rette linjene på kontrollbussen, slik at minnet får beskjed om at det skal leses og at adressen på adressebussen er gyldig
2. DBR henter data fra databussen. Dermed ligger den ene operanden i DBR (og den andre ligger i AC fra før).
3. Den ene operanden må kopieres til Y. Vi velger å flytte innholdet av DBR til Y ved å åpne for dataflyt mellom de to registrene.
4. Det utføres en addisjon med ALU. Det ene tallet ligger i Y. Det andre tallet hentes fra AC ved at det åpnes for dataflyt mellom AC og ALUens inngang. Resultatet havner i Z-registeret.
5. Resultatet skal ligge i AC nå instruksjonen er ferdig, derfor åpnes det for dataflyt mellom Z og AC.

Utføringssyklusen for ADD-instruksjonen tok altså 5 klokkesykluser. Ulike instruksjoner trenger ulikt antall klokkepulser avhengig av hvor komplisert instruksjonen er.

Kort oppsummering

Vi har sett på CPUens virkemåte og repetert det vi har lært om delsyklusene til instruksjons-syklusen. Vi har introdusert de forskjellige delene av CPUen. Deretter så vi hvordan instruksjonene blir brutt ned i mikrooperasjoner. Disse mikrooperasjonene blir utført under kontroll av kontrollenheten. Hver mikrooperasjon blir utført i løpet av en klokkesyklus. Mikrooperasjoner er operasjoner på svært lavt nivå.

Senere i kurset skal vi se på kontrollenheten, og hvordan den er bygget opp.