

Institutt for datateknikk og informasjonsvitenskap

## Eksamensoppgave i TDT4160 datamaskiner og digitalteknikk

Faglig kontakt under eksamen: **Gunnar Tufte**

Tlf.: 97402478

**Eksamensdato: 17. august 2017**

**Eksamenstid (fra-til): 9:00–13:00**

**Hjelpemiddelkode/Tillatte hjelpemidler:** D: Ingen trykte eller håndskrevne hjelpemidler tillatt.

Bestemt, enkel kalkulator tillat.

**Annen informasjon:**

**Målform/språk: Bokmål**

**Antall sider (uten forside): 10**

**Antall sider vedlegg: 2**

**Informasjon om trykking av eksamensoppgave**

Originalen er:

**1-sidig**  **2-sidig**

**sort/hvit**  **farger**

**skal ha flervalgskjema**

**Kontrollert av:**

\_\_\_\_\_  
Dato

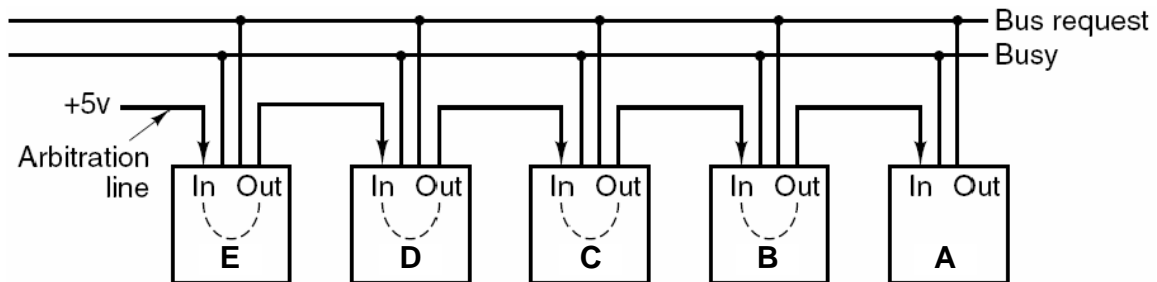
\_\_\_\_\_  
Sign

Oppgave 1 Oppstart, litt av hvert (20%)

a)

Figur 1 viser en skisse av en mulig bussarbitreringsmekanisme. Ut fra informasjonen i figur 1 svar på følgende spørsmål:

- i) Hvilken enhet vil få bussen hvis enhet C og D gjør en «Bus request» samtidig? Forklar kort.
- ii) Er dette en sentralisert eller en desentralisert arbitreringsmekanisme? Forklar kort.



Figur 1 Bussarbitrering.

b)

Angi endring i aksesstid, lagringskapasitet og pris (bit/kr) når man går nedover i lagerhierarkiet (memory hierarchies), fra CPU-register, hurtigbuffer til hovedminne (main memory).

c)

Hva er «latency hiding», og hvorfor har dette fått økt oppmerksomhet ved økt CPU-ytelse? Forklar kort.

d)

Hva er en «interrupt vector»? Forklar kort.

e) De tre instruksjonene vist under (I1, I2 og I3) er del av et større program. Er det avhengigheter (true dependence) i koden under? Forklar kort.

I1: ADD R1, R2, R3; ( $R1 = R1 + R3$ ,  $Rn$  Register nummer)

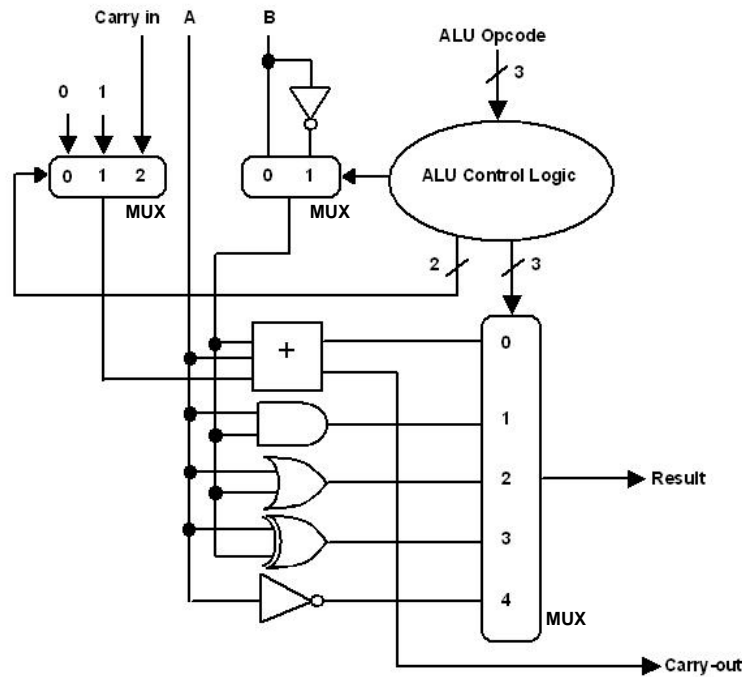
I2: SUB R4, R1, R5; ( $R4 = R1 + R5$ )

I3: AND R0, R7, R9; ( $R0 = R7 \text{ and } R9$ )

Oppgave 2 Digitalt Logisk Nivå (20% (a: 5%, b: 7.5% og c: 7.5%))

a)

Figur 2 viser en skisse av innholdet i en ALU. Det er to datainnganger (A og B), én datautgang (Result), én Carry-inngang og én Carry-utgang. ALU-en har tre kontrollinnganger for å bestemme funksjon. Internt er det komponenter for aritmetikk og logikk, disse kan repliseres til ønsket bitbredde, tre multiplexere og kontrolllogikk. ALU-en bruker en klokkeperiode på å utføre en operasjon.

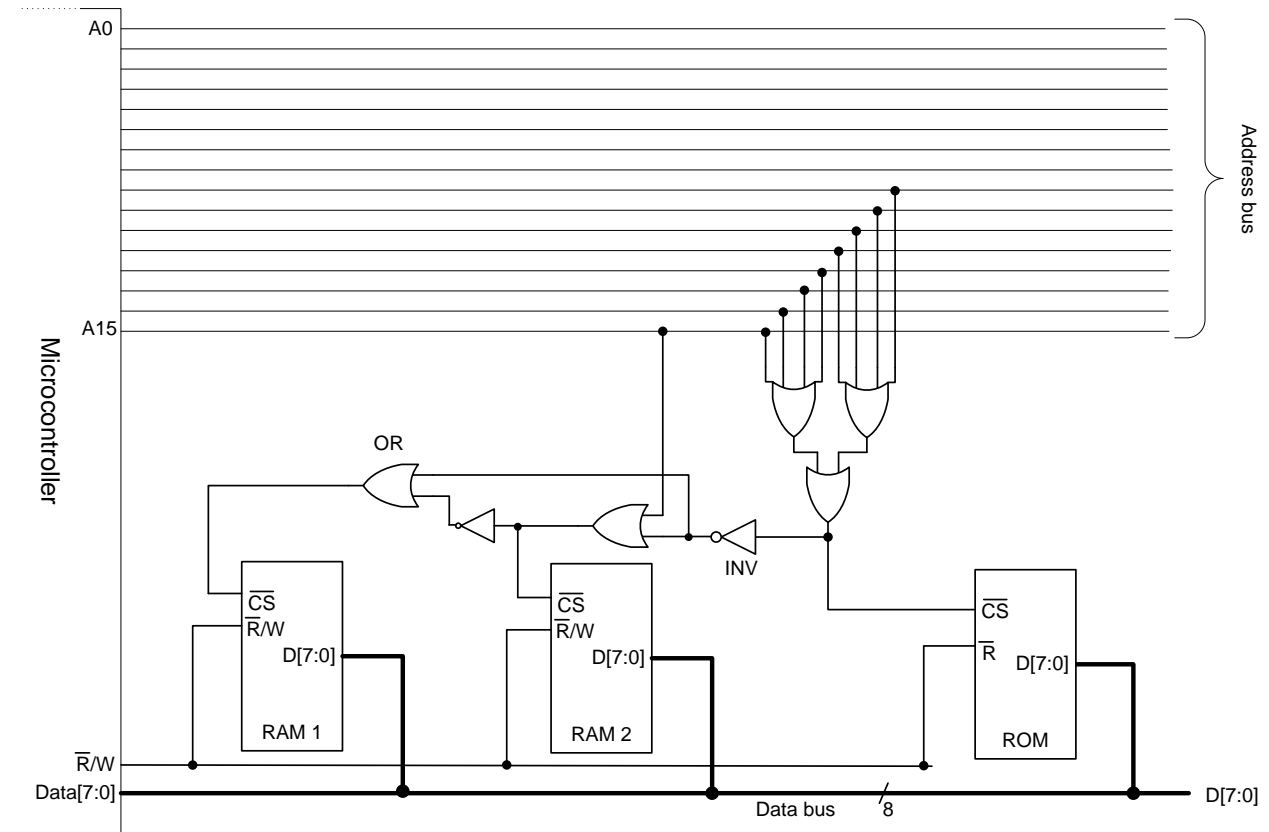


Figur 2 ALU-skisse.

Hvor mange klokkeperioder er nødvendig for å utføre en subtraksjon ( $\text{Result} = \text{operand1} - \text{operand2}$ )? Forklar hvilke ALU operasjoner som må utføres for en subtraksjonen. Du kan fritt velge hvilken datainngang (A eller B) operandene tilordnes. Det er ikke nødvendig å spesifisere eventuelle mellomlagringer i spesifikke register, bruk tilgjengelig informasjon og oppgi eventuelle forutsetninger.

b)

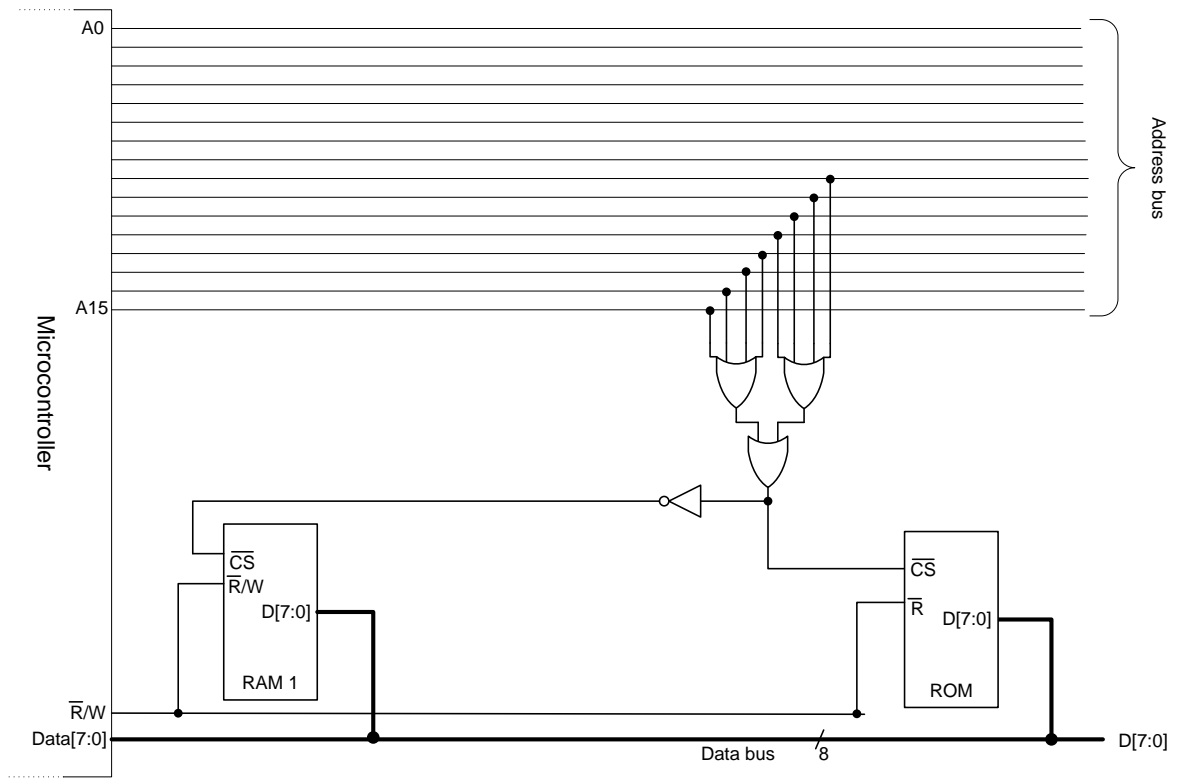
I et innvevd system (embedded system) er det benyttet én mikrokontroller, to RAM-brikker på 32k byte og én ROM-brikke på 16k byte. Figur 3 viser det eksterne bussgrensesnittet med adressedekodingslogikk. Alle enhetene benytter et aktivt lavt (logisk "0") CS (Chip Select)-signal. Alle enhetene benytter 8-bit data.



Figur 3 Adresse decoding 1.

I et forsøk på å spare komponenter blir systemet endret til å bare bruke en RAM-brikke på 64k byte. Det nye systemet er vist i figur 4.

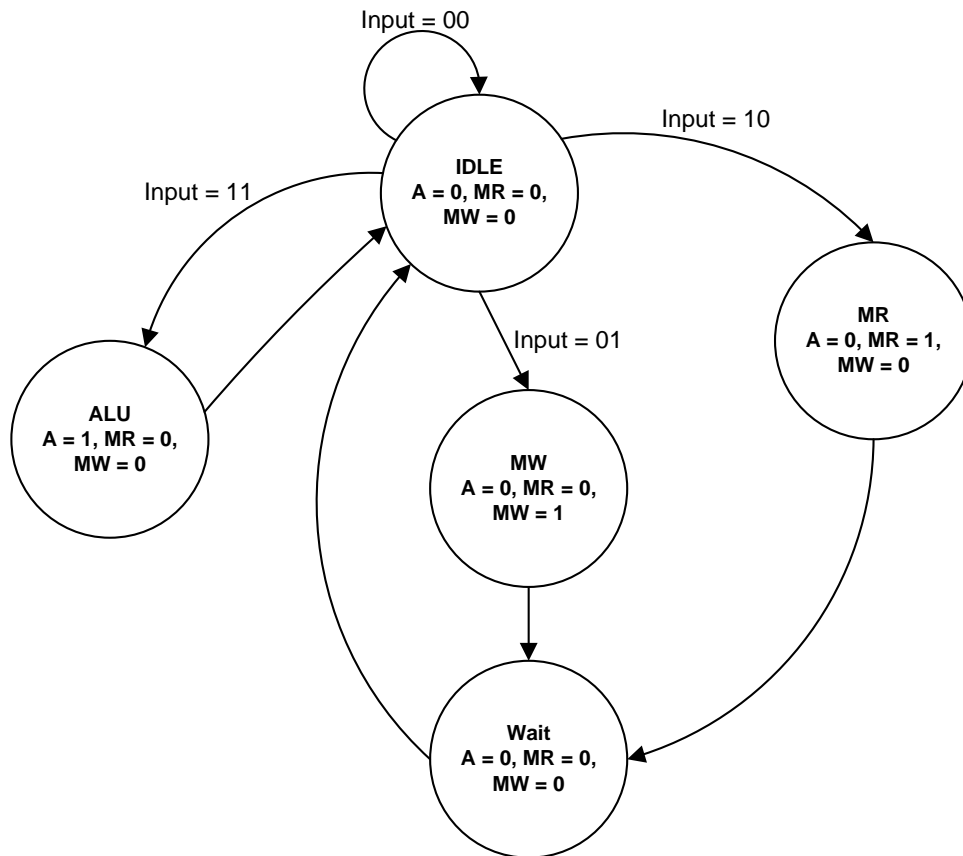
- i) Er de to systemene vist i figur 3 og figur 4 kompatible (likt adresseområde)? Forklar.
- ii) Vil et program skrevet for systemet vist i figur 3 virke uten endringer på systemet vist i figur 4? Forklar kort hvorfor/hvorfor ikke.



Figur 4 Adresse decoding 2.

c)

Figur 5 viser tilstandsdiagrammet for en FSM som brukes i kontrollogikken i en CPU. Det er to bit input fra dekodingslogikken (Input), og tre utgangssignal (output): A (enable for ALU), MW (memory write signal) og MR (memory read signal).



Figur 5 Finite State Machine (FSM)

i) Fullfør transisjonstabellen på følgende format:

Current state	Nxt state Input=00	Nxt state Input=01	Nxt state Input=10	Nxt state Input=11	Output A	Output MR	Output WR
Idle	Idle	MW	MR	ALU	0	0	0

ii) Hva er minimum størrelse på stateregisteret (antall flip-flops)?

*Oppgave 3 Mikroarkitektur og mikroinstruksjoner (20% (a: 7.5%, b: 7.5% og c: 5%))*

Bruk vedlagte diagram i figur 10, figur 11, figur 12 og figur 13 for IJVM til å løse oppgavene.

a)

Forklar kort funksjonen til de forskjellige feltene i MIR (Addr, J, ALU, C; M og B).

b)

For mikroarkitekturen i Figur 9. Angi mikroinstruksjon(er) for å utføre en NAND funksjon mellom innholdet i registerene LV og TOS. Resultatet skal lagres i OPC-registeret (OPC = LV NAND TOS).

Se vekk fra Addr- og J-feltet i mikroinstruksjonsformatet. Angi korrekte bit for ALU, C, Mem og B gitt i figur 10.

c)

Hvilke parameter bestemmer klokkefrekvensen denne versjonen av IJVM kan operere på?

*Oppgave 4 Instruksjonssett arkitektur (ISA) (20% (a: 5%, b: 5% og 10% på c))*

Letni X68 er en svært enkel prosessor. Letni X68 har én «load», én «store», åtte ALU-instruksjoner og noen spesialinstruksjoner, inkludert NOP-instruksjonen og tre flytkontrollinstruksjoner (flow control instructions). Instruksjonsformatet for instruksjonene er vist i figur 6, figur 7 viser instruksjonssettet. Alle register og busser er 32-bit. Det er 32 generelle register tilgjengelig. Prosessoren har en Harvard-arkitektur. Bruk figur 6 og figur 7 til å løse oppgaven.

a)

Hva betyr det at Letni X68 har en Harvard-arkitektur?

b)

Hvilke instruksjonsformat(er) benytter Letni X68?

c)

R0 har følgende verdi: 0x0000 FFFF, R8 har følgende verdi: 0xFFFF 0000, R11 har følgende verdi: 0x0000 0000 og R12 har følgende verdi: 0x0000 0003. I dataminnnet ligger følgende data fra adresse 0xFFFF 0000:

Adresse	Data
0xFFFF 0000:	0x00 00 00 55
0xFFFF 0001:	0x00 AA 00 00
0xFFFF 0002:	0x00 00 00 AA
0xFFFF 0003:	0x00 00 55 00
0xFFFF 0004:	0x00 00 00 00

Følgende psaudokode er en del av et større program. Kodesnutten starter på adresse 0000 FFFF i programminnet. Svar på spørsmålet ut fra tilgjengelig informasjon.

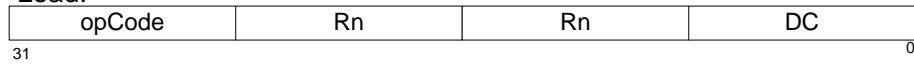
```
0x0000 FFFF: LOAD R1, R8;
0x0001 0000: ADD R11, R1, R11;
0x0001 0001: INC R8, R8;
0x0001 0002: DEC R12, R12
0x0001 0003:BNZ R0;
0x0001 0004      :
```

Forklar hva som skjer i koden. Hvilken verdi vil R11 ha etter at koden har kjørt (PC = 0001 0004)?

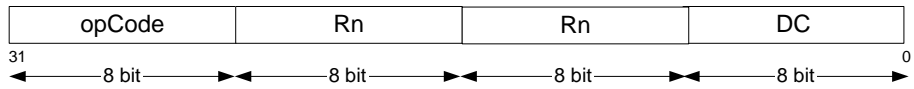


Load/store:

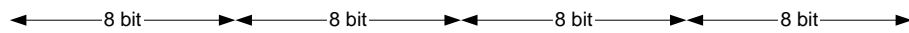
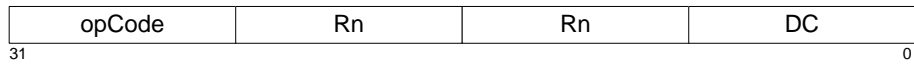
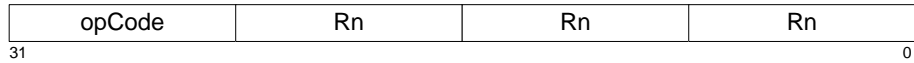
Load:



Store:

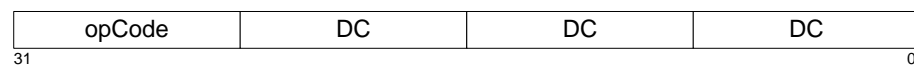


ALU:



Special:

NOP:



MOVC:

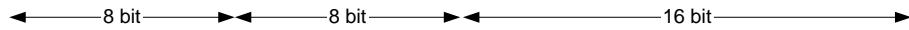


CP:



Flow control:

BZ/BNZ



RT



Rn: any user register, R0 - R31

DC: Don't care: any data memory location

Figur 6 Instruction format Letni X68

## Instructions set:

**LOAD:** Load data from memory.

load  $R_i, R_j$  Load register  $R_i$  from memory location in  $R_j$ .

**STORE:** Store data in memory.

store  $R_i, R_j$  Store register  $R_i$  in memory location in  $R_j$ .

**ALU:** Data manipulation, register-register operations.

**ADD**  $R_i, R_j, R_k$  ADD,  $R_i = R_j + R_k$ . Set Z-flag if result =0.

**NAND**  $R_i, R_j, R_k$  Bitwise NAND,  $R_i = \overline{R_j \cdot R_k}$ . Set Z-flag if result =0.

**OR**  $R_i, R_j, R_k$  Bitwise OR,  $R_i = R_j + R_k$ . Set Z-flag if result =0.

**INV**  $R_i, R_j$  Bitwise invert,  $R_i = \overline{R_j}$ . Set Z-flag if result =0.

**INC**  $R_i, R_j$  Increment,  $R_i = R_j + 1$ . Set Z-flag if result =0.

**DEC**  $R_i, R_j$  Decrement,  $R_i = R_j - 1$ . Set Z-flag if result =0.

**MUL**  $R_i, R_j, R_k$  Multiplication,  $R_i = R_j * R_k$ . Set Z-flag if result =0.

**CMP**,  $R_i, R_j$  Compare, Set Z-flag if  $R_i = R_j$

**Special:** Misc.

**CP**  $R_i, R_j$  Copy,  $R_i < -R_j$  (copy  $R_j$  into  $R_i$ )

**NOP** Waste of time, 1 clk cycle.

**MOVC**  $R_i, \text{constant}$  Put a constant in register  $R_i = C$ .

**Flow control:** Branch.

**BZ**,  $R_i$  Conditional branch on zero (Z-flag = 1) ,  $PC = R_i$ .

**BNZ**,  $R_i$  Conditional branch on non zero (Z-flag = 0),  $PC = R_i$ .

**RT** Return, return from branch.

$R_i, R_j$  and  $R_k$ : Any user register.

DC: Don't care.

Oppgave 5 Ytelse (20 (20% (a: 5%, b: 10%, og 5% på c))

a)

En prosessor har et minnesystem med RAM og et nivå med hurtigbuffer (cache). Minnesystemet har følgende egenskaper:

RAM: aksesstid: 100 ns

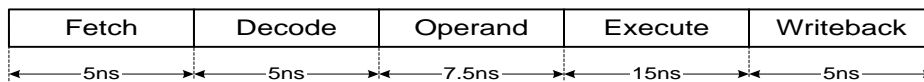
Cache: aksesstid 10 ns

Trefforholdstall (Hit ratio) på 80 %

Hva betyr det at systemet har en Trefforholdstall (Hit ratio) på 80 %? Forklar kort

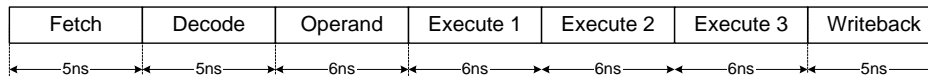
b)

1) Figur 8 viser samlebandet (pipeline) for en prosessor. Det er fem steg. Hvor lang tid vil det ta å utføre et program på 100 instruksjoner? Gå ut fra at det ikke er noen avhengigheter eller hopp i koden.



Figur 8 5 stage pipeline.

2) For å øke ytelsen blir operand og execute-endret og utvidet til fire steg som vist i figur 9. Hvor lang tid vil det ta å utføre programmet på denne versjonen av prosessoren?



Figur 9 7 stage pipeline.

c) Er økt ILP hoveddrivkraften for å utvide antal kjerner i en Chip Multiprocessor? Forklar kort hvorfor/hvorfor ikke.

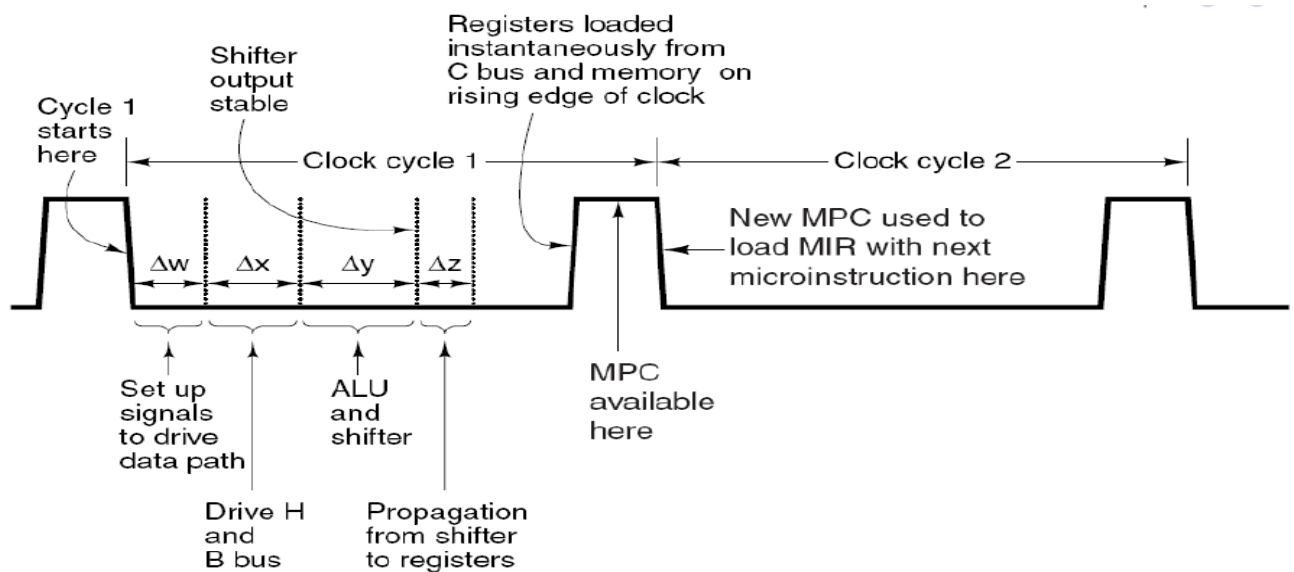
# Vedlegg IJVM



$F_0$	$F_1$	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B - A
1	1	0	1	1	0	B - 1
1	1	1	0	1	1	-A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	-1

SLR1	SLL8	Function
0	0	No shift
0	1	Shift 8 bit left
1	0	Shift 1 bit right

Figur 12 ALU functions.



Figur 13 Timing diagram.