Norwegian University of Science and Technology
Engineering
The Department of Computer and Information Science

TDT4160
DATAMASKINER OG DIGITALTEKNIKK
EKSAMEN

2. DESEMBER, 2015, 09:00–13:00

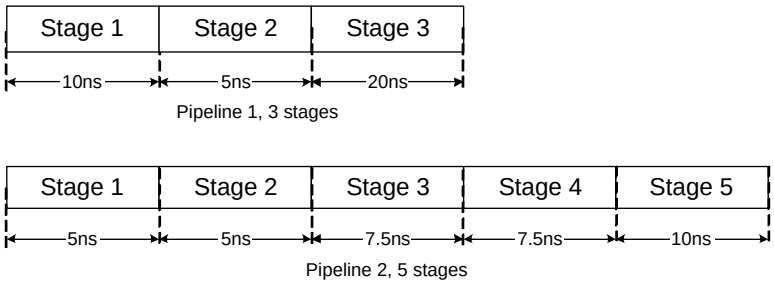
Kontakt under eksamen:
Gunnar Tufte 974 02 478

Tillatte hjelpemidler:
D.
Ingen trykte eller håndskrevne hjelpemidler er tillat.
Enkel godkjent kalkulator er tillatt.

Målform:
Bokmål

OPPGAVE 1: OPPSTART, LITT AV HVERT (25 %)

a. En ISA har tre alternative mikroarkitekturer. Alternativ 1 er utan samlebånd (pipeline) med en klokkeperiode på $40ns$. Alternativ 2 er med et samlebånd med tre trinn. Alternativ 3 er med et samlebånd på 5 trinn. Alternativ 2 og 3 er skissert i figur 1. Er det mulig å øke klokkefrekvensen i mikroarkitekturene i alternativ 2 og 3? Oppgi klokkefrekvens for de to alternativene.



Figur 1: Samleband for alternativ 2 og 3.

b. En prosessor har et enkelt minnesystem med et hoveddminne (RAM) med aksesstid på $0.5\mu s$. I et forsøk på å skalere opp systemet og øke ytelsen ønsker man å doble minnestørrelsen og minske gjennomsnittlig aksesstid. I et forsøk blir hovedminnet byttet ut med et dobbelt så stort minne med aksesstid på $2\mu s$ og et nivå hurtigbuffer (cache). Hurtigbuffer har en aksesstid på $0,025\mu s$ og trefforholdstall for systemet (hit ratio) er 90 %

   Hva er gjennomsnittlig aksesstid for det nye minnesystemet (med et nivå hurtigbuffer)?

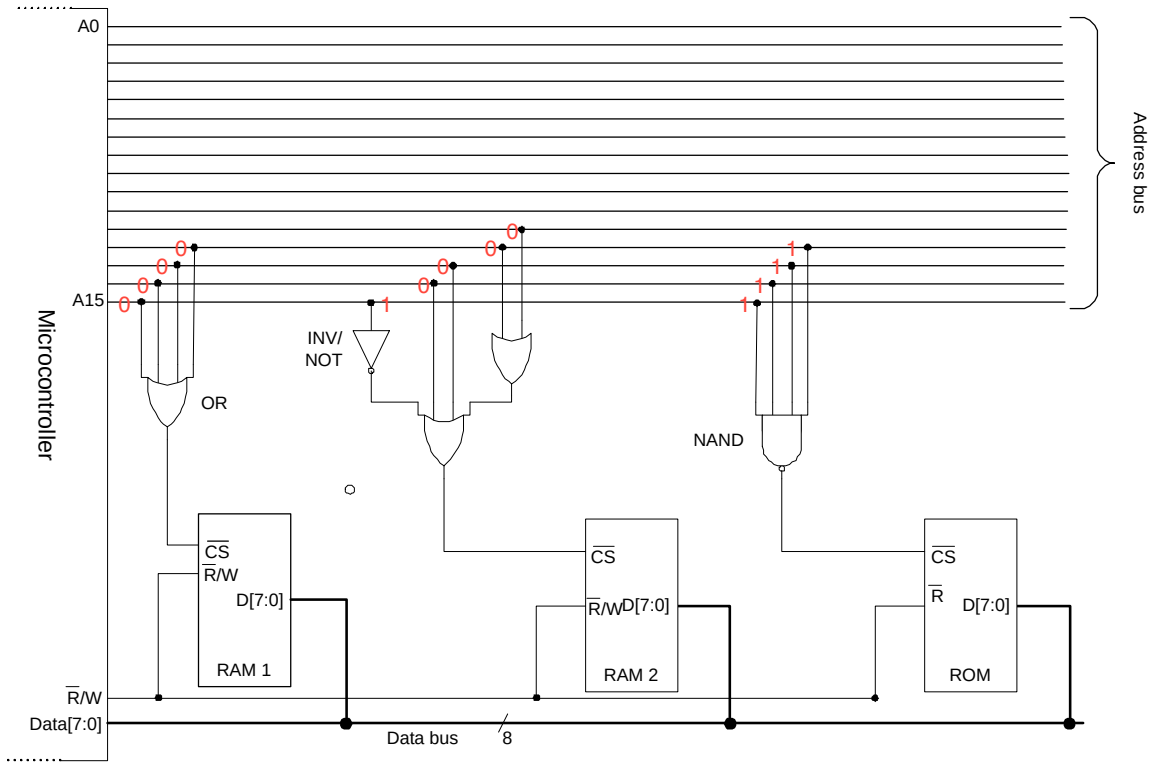
c. Hva kjennetegner en superscalar prosessor? Forklar kort.

d. Hva kjennetegner en SIMD-prosessor? Forklar kort.

e. Er grad av ILP definert på ISA-nivå?

OPPGAVE 2: DIGITALT LOGISK NIVÅ (25 % (10 % PÅ A, 5 % PÅ B OG 10 % PÅ C))

I et innvevd system (embedded system) benyttes det en mikrokontroller. Figur 2 viser det eksterne bussgrensesnittet for mikrokontrolleren. Det er en ROM-brikke for program og to RAM-brikker. Alle enhetene benytter et aktivt lavt (logisk "0") CS (Chip Select)-signal. RAM-brikkene har en aksesstid på 0.1μs ROM-brikken har en aksesstid på 1μs.



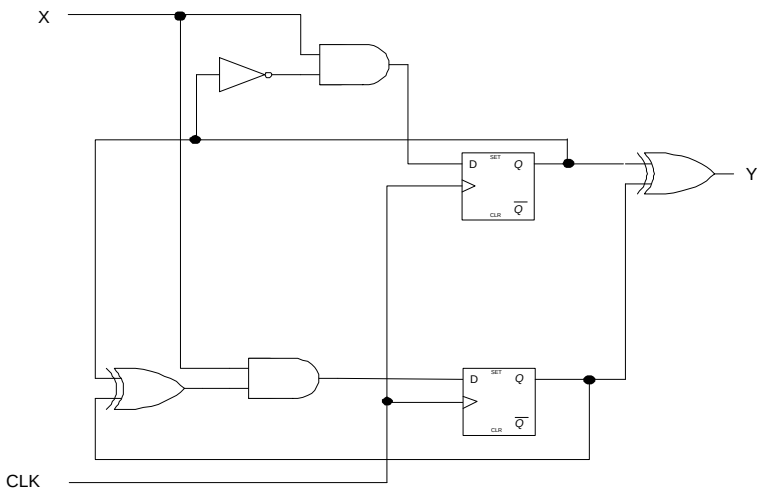
Figur 2: Address decoding.

a. i) Tegn minnekart for systemet.

   ii) Er det mulig å utvide med mer RAM?.

b. Dette er en einsykel (single cycle) prosessor uten hurtigbuffer eller instruksjonskø. Systemet har en klokke på 500 000hz. Prosessoren utfører nå en instruksjon hver klokkeperiode. Er det mulig å øke klokkefrekvensen til systemet til 2 000 000hz og fortsatt kunne utføre en instruksjon hver klokkeperiode? Forklar kort.

c. Figur 3 viser en FSM.



Figur 3: FSM.

i) Er dette en Moore eller Mealy FSM? Forklar.
ii) Fullfør tabellen vist under. Bruk formatet som er vist.

Q0 Q1	Q0(nxt) Q1(nxt)	
	X = 0	X = 1
0 0		
1 0		
0 1		
1 1		

Figur 4: FSM tabell format.

iii) Når er utgangen Y = "1"?

OPPGAVE 3: MIKROARKITEKTUR OG MIKROINSTRUKSJONER (25 % (5 % PÅ A, 10 % PÅ B OG C))

Bruk vedlagte diagram i figur 6, figur 7, figur 8 og figur 9 for IJVM til å løse oppgavene.

a. i) Er det noen av registrene i IJVM som ikke er tilgjengelig for programmereren?
ii) Hvilken funksjon har MAR-registeret og MDR-registeret? Forklar kort.

b. i) Lag mikroinstruksjon(er) som utfører funksjonen: kopier TOS til MAR (MAR = TOS) for mikroarkitekturen i figur 6.
ii) Lag mikroinstruksjon(er) som utfører funksjonen: summerer innholdet i LV, MDR og PC til PC (PC = LV + MAR + PC) for mikroarkitekturen i figur 6. Hvordan vil LV og MAR bli påvirket av løsningen din?

   Du trenger ikke ta hensyn til Addr- og J-feltene. Oppgi korrekt bit-verdi for ALU-, C-, Mem- og B-feltene. – Se figur 7.

c. IJVM-registrene i figur 6 er satt til følgende verdiar:

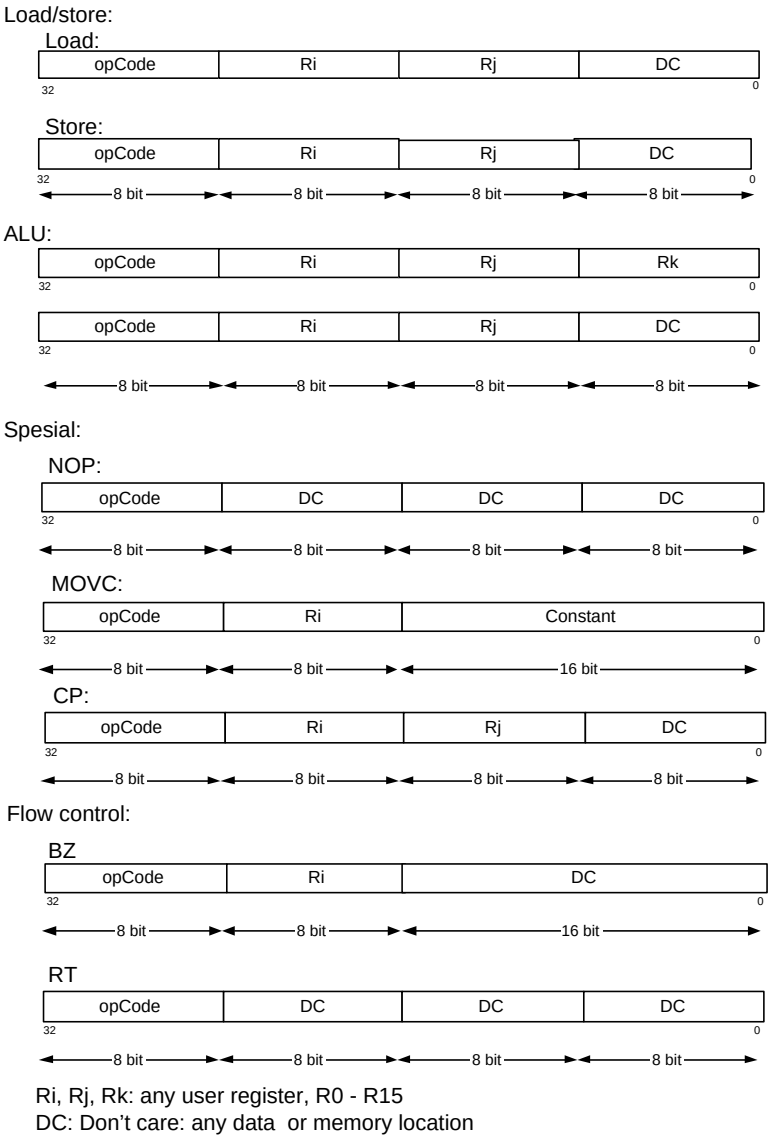
   ”SP”: hex(0001),
   ”LV”: hex(0002),
   ”CPP”: hex(0003),
   ”TOS”: hex(0004),
   ”OPC”: hex(0005),
   ”H”: hex(FF0A).

   Hva er innholdet i H-registeret og OPC-registeret etter at følgende mikroinstruksjoner har blitt utført? Oppgi svaret i hex-format.

   1: ALU: 010100, C: 100000000, Mem: 000 og B: 0110
   2: ALU: 011100, C: 101000000, Mem: 000 og B: 0111
   3: ALU: 001100, C: 001000000, Mem: 000 og B: 1000

OPPGAVE 4: INSTRUKSJONSSETT ARKITEKTUR (ISA)(25 % ((5 % PÅ A, B,C OG 10 % PÅ D))

En avært enkel prosessor har en load, en store, 8 ALU-instruksjoner og noen spesialinstruksjoner, inkludert NOP-instruksjonen og to flytkontrollinstruksjoner (flow control instructions). Instruksjonsformatet for instruksjonene er vist i figur 5. Alle register og bussar er 32-bit. Det er 16 generelle register tilgjengelig. Prosessoren har en Harvard architecture.



Figur 5: Instruction formats.

Instructions set:

LOAD: Load data from memory.

load Ri, Rj Load register Ri from memory location in Rj.

STORE: Store data in memory.

store Ri, Rj Store register Ri in memory location in Rj.

ALU: Data manipulation, register–register operations.

ADD Ri, Rj, Rk ADD, $Ri = Rj + Rk$. Set Z-flag if result =0.
NAND Ri, Rj, Rk Bitwise NAND, $Ri = \overline{Rj \cdot Rk}$. Set Z-flag if result =0.
OR Ri, Rj, Rk Bitwice OR, $Ri = Rj + Rk$. Set Z-flag if result =0.
INV Ri, Rj Bitwice invert, $Ri = \overline{Rj}$. Set Z-flag if result =0.
INC Ri, Rj Increment, $Ri = Rj + 1$. Set Z-flag if result =0.
DEC Ri, Rj Decrement, $Ri = Rj - 1$. Set Z-flag if result =0.
MUL Ri, Rj, Rk Multiplication, $Ri = Rj * Rk$. Set Z-flag if result =0.
CMP, Ri, Rj Compare, Set Z-flag if $Ri = Rj$

Special: Misc.

CP Ri, Rj Copy, $Ri < -Rj$ (copy Rj into Ri)
NOP Waste of time, 1 clk cycle.
MOVC Ri, constant Put a constant in register $Ri = C$.

Flow control: Branch.

BZ, Ri Conditional branch on zero, $PC = Ri$.
RT Return, return from branch.

Ri, Rj and Rk: Any user register.
DC: Don't care.

a. Hva slags type instruksjoner er:
   i) BZ?
   ii) RT?

b. Hva slags type adresseringsmodi (addressing modes) bruker:
   i) MOVC?
   ii) CP?

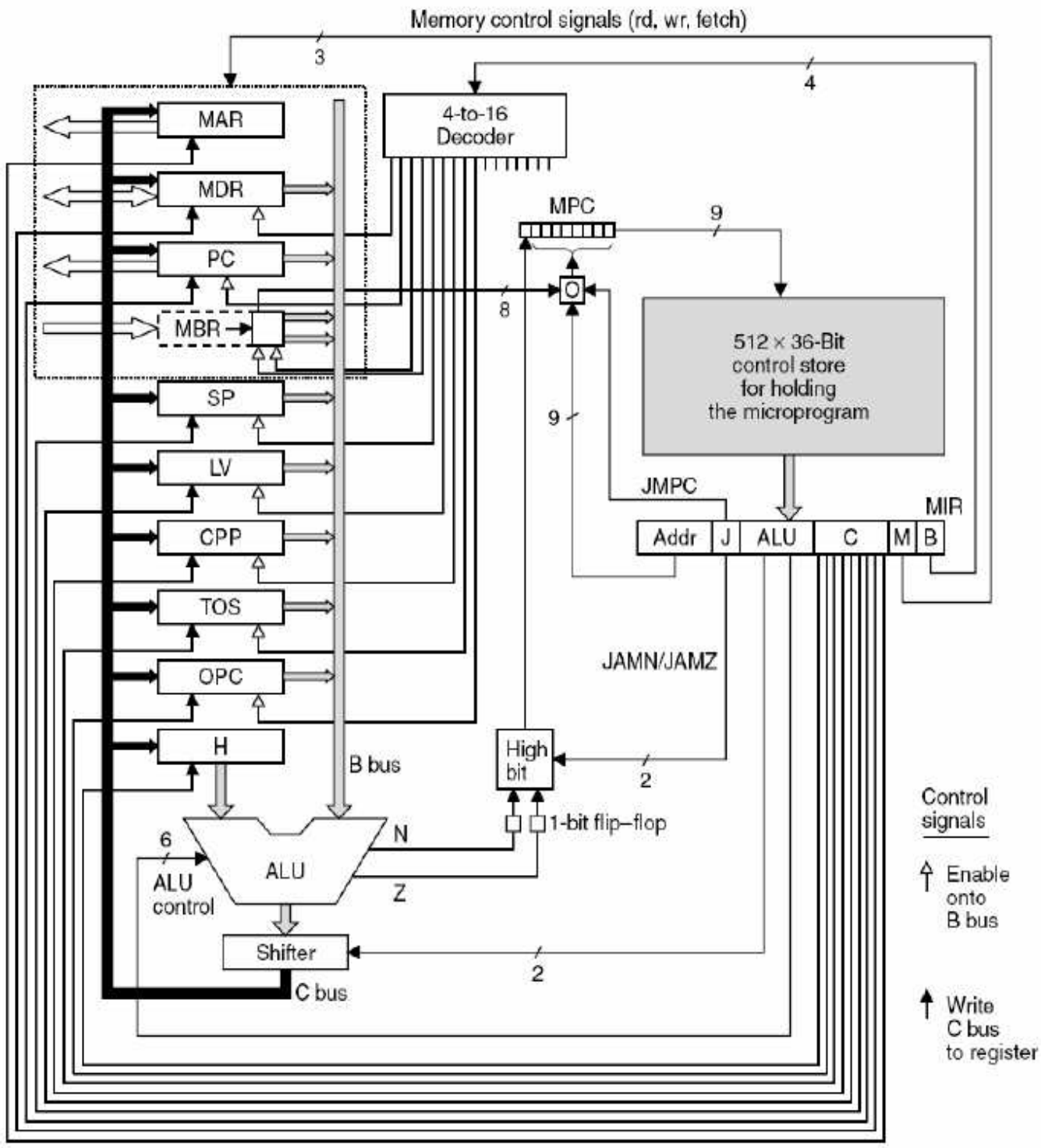
c. Kan LOAD- og STORE-instruksjonene adressere hele adresserommet (32-bit adressebusser)?

d. Følgende psaudokode er en del av et større program. Svar på spørsmålene ut i fra tilgjengelig innformasjon.

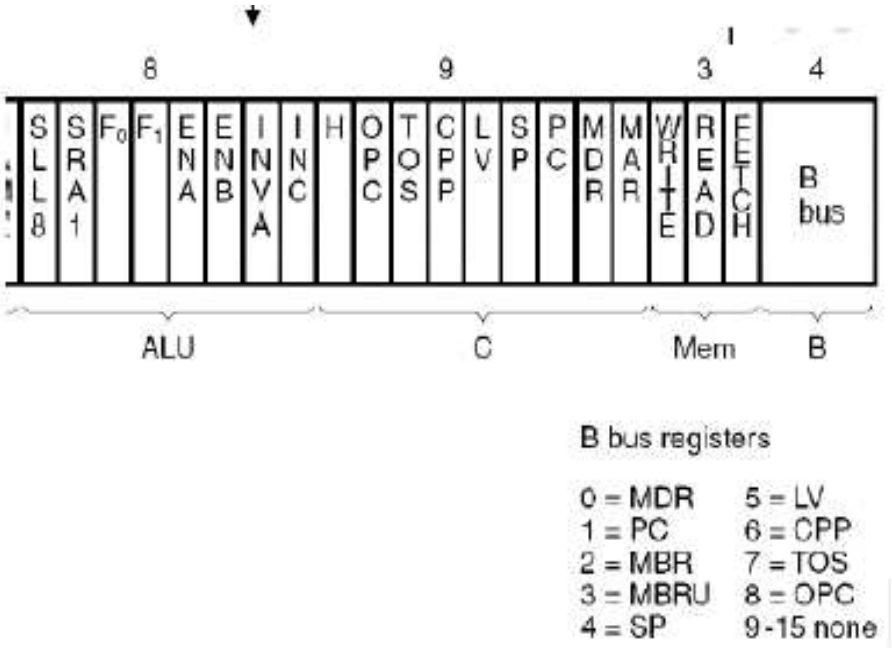
   - MOVC R1, 0x00 55;
   - MOVC R2, 0x00 AA;
   - STORE R1, R2;
   - ADD R1, R1, R3;
   - STORE R1, R3;
   - LOAD R1, R2
   - INV R3, R3;
   - CMP R1, R2;
   - BZ R2;

   i) Hva slags adresse vil BZ hoppe til hvis betingelsene for et hopp er til stede?
   ii) Vil psaudokode føre til at Z-flagget blir aktivert slik at betingelsene for et hopp er til stede? Begrunn svaret.

IJVM appendix



Figur 6: Block diagram (IJVM).



Figur 7: Microinstruction format (IJVM).

Answer Key for The Exam

OPPGAVE 1: OPPSTART, LITT AV HVERT (25 %)

a. En ISA har tre alternative mikroarkitekturer. Alternativ 1 er utan samlebånd (pipeline) med en klokkeperiode på $40ns$. Alternativ 2 er med et samlebånd med tre trinn. Alternativ 3 er med et samlebånd på 5 trinn. Alternativ 2 og 3 er skissert i figur 1. Er det mulig å øke klokkefrekvensen i mikroarkitekturene i alternativ 2 og 3? Oppgi klokkefrekvens for de to alternativene.

Answer: Alternativ 1: Klokkefrekvens $= 1/t = 1/40ns$ f = 25Mhz

I dei to andre er det eit samlebånd. I eit samlebånd er det det treigaste trinne som angir klokka. Alle trinn må bli ferdig innan for ein klokkeperiode.

Alternativ 2: Klokkefrekvens $= 1/t = 1/20ns$ f = 50Mhz
Alternativ 3: Klokkefrekvens $= 1/t = 1/1ns$ f = 100Mhz
Så klokke frekvensen kan aukast ved innføring av samleband i denne mikroarkitekturen. I alternativ 2 og 3 kan klokkefrekvensen aukast til (max) forsinkelse i treigaste trinn.

b. En prosessor har et enkelt minnesystem med et hoveddminne (RAM) med aksesstid på $0.5\mu s$. I et forsøk på å skalere opp systemet og øke ytelsen ønsker man å doble minnestørrelsen og minske gjennomsnittlig aksesstid. I et forsøk blir hovedminnet byttet ut med et dobbelt så stort minne med aksesstid på $2\mu s$ og et nivå hurtigbuffer (cache). Hurtigbuffer har en aksesstid på $0,025\mu s$ og trefforholdstall for systemet (hit ratio) er 90 %

Hva er gjennomsnittlig aksesstid for det nye minnesystemet (med et nivå hurtigbuffer)?

Answer: $2s$; mean access time = $c + (1 - h) * m)$; $0.025 + (1 - 0.9) * 2) = 0.225$
Viser at det er mulig å få eit særs effektivt minnesystem ved å innføre hurtigbuffer. Det nye systemet er raskare sjølv om hovedminne har auka akesstid med faktor 4.

c. Hva kjennetegner en superscalar prosessor? Forklar kort.

Answer: Innfører fleire av ein eller fleire einingar i ein prosessor med samleband, for eksempel felles fetch og decode, fleire ALUar. Sjå (f.eks) figur 2.6 i boka.

d. Hva kjennetegner en SIMD-prosessor? Forklar kort.

Answer: Single Instruction stream Multiple Data streams.
Parallelprosessorarklasse der multipleprosessorar utfører samme instruksjon på forskjellige data. Typisk array prosessorar, graffik er eit egna datasett for ein slik type arkitektur. (klasseifisert i Flynn's taxonomyfor datamaskin arkitektur).

e. Er grad av ILP definert på ISA-nivå?

Answer: ILP: Instruktion Level Paralellism. Nei, ILP angir mengd instruksjonar som utføres samtidigt, f.eks pipeliner med fleire steg (djubde) vil ein få høgare ILP, enn med få steg. MEN ILP er ikkje definert på ISA-nivå, ISA definerar instruksjonar logisk. Ein ISA kan ha fleire alternative mikroarkitekturar med forskjelig ytelse (med forskjellig ILP).

OPPGAVE 2: DIGITALT LOGISK NIVÅ (25 % (10 % PÅ A, 5 % PÅ B OG 10 % PÅ C))

I et innvevd system (embedded system) benyttes det en mikrokontroller. Figur 2 viser det eksterne bussgrensesnittet for mikrokontrolleren. Det er en ROM-brikke for program og to RAM-brikker. Alle enhetene benytter et aktivt lavt (logisk "0") CS (Chip Select)-signal. RAM-brikkene har en aksesstid på $0.1\mu s$ ROM-brikken har en aksesstid på $1\mu s$.

a. i) Tegn minnekart for systemet.
   ii) Er det mulig å utvide med mer RAM?.

Answer: i)
RAM 1: 0 - 0FFF
RAM 2: 8000 - 87FF
ROM: F000 - FFFF.
Det skal teiknast minnekart.
ii) Det er to ledige områder som kan brukast ved behov. (8800 - EFFF og 1000 - 7FFF).
Mengd RAM kan utvidast.

b. Dette er en einsykel (single cycle) prosessor uten hurtigbuffer eller instruksjonskø. Systemet har en klokke på 500 000hz. Prosessoren utfører nå en instruksjon hver klokkeperiode. Er det mulig å øke klokkefrekvensen til systemet til 2 000 000hz og fortsatt kunne utføre en instruksjon hver klokkeperiode? Forklar kort.

Answer: Nei, ROM-brikken kan kunn følgje med oppmot max 1Mhz (sjølv om prosessoren kan handtere auka klokke). Ved 2Mhz vil aksesstida møtte være min. $0.5\mu s$ for å klare å halde kravet om ein instruksjon kvar klokkeperiode.

c. Figur 3 viser en FSM.

i) Er dette en Moore eller Mealy FSM? Forklar.
ii) Fullfør tabellen vist under. Bruk formatet som er vist.

iii) Når er utgangen Y = "1"?

Answer: i) Moore, kunn state blir brukt til dekoding for utgangssignal. (ingen kopling frå inngangar til dekodingen av state (q1 og Q0)
ii) Må definere kva som må til for at D0 og D1 skal være "1". Velger øverste som D0/Q0, får då:
D0 = X and not(Q0)
D1 = X and (Q1 xor Q0)
Q0 = D0 og Q1 = D1

Utgangen gitt av Q0 og Q1:
Y = Q1 xor Q0.

iii) Y = Q1 xor Q0, ved state 01 og 10. X kan sjøast som f.eks reset signal (synkront). Ved X = 0 går FSM tilbake til state 00.

OPPGAVE 3: MIKROARKITEKTUR OG MIKROINSTRUKSJONER (25 % (5 % PÅ A, 10 % PÅ B OG C))

Bruk vedlagte diagram i figur 6, figur 7, figur 8 og figur 9 for IJVM til å løse oppgavene.

a. i) Er det noen av registrene i IJVM som ikke er tilgjengelig for programmereren?
ii) Hvilken funksjon har MAR-registeret og MDR-registeret? Forklar kort.

Answer: i) Register i datapath: MBR-register, kan ikkje skrivast til, ingen buss tilkopling for skriving, kunn loadfrå prog. minne. MAR kan ikkje lesast. For programerarar er det då ikkje mulig å bruke desse registera direkte (uansett om det er IJVM ISA eller ein alternativ ISA definert i control store. Ellers MIR og MPC (rett men dei er interne register i CTRL-einingen (ikkje del av ISA) ingen minus viss dei ikkje er nevnt).
ii) MAR: Memory Adr. Reg. Innheld peikar til minnelokasjon (eksternt data minne), MDR: Mem. Data Reg. inneheld data frå minne lokasjon peika på av MAR ved read. Inneheld data til minnelokasjon (peika på av MAR) ved write.

b. i) Lag mikroinstruksjon(er) som utfører funksjonen: kopier TOS til MAR (MAR = TOS) for mikroarkitekturen i figur 6.
ii) Lag mikroinstruksjon(er) som utfører funksjonen: summerer innholdet i LV, MDR og PC til PC (PC = LV + MAR + PC) for mikroarkitekturen i figur 6. Hvordan vil LV og MAR bli påvirket av løsningen din?

Du trenger ikke ta hensyn til Addr- og J-feltene. Oppgi korrekt bit-verdi for ALU-, C-, Mem- og B-feltene. – Se figur 7.

Answer: i)
ALU: B, C: MAR, B: TOS
ALU: 010100, C: 000000001, B: 0111 (7).
ii)
Her er det ein typo i eksamensette. Skal være i parantesen: (PC = LV + MDR + PC). Det står rett i txt beskrivelsen av instruksjonen. Gir og rett for svar som skjønnar at ein må dele i fleire mikroinstruksjonar og mellomlagre i H. Også rett viss nokon seier at dette går ikkje fordi MAR ikkje kan legjast ut på B-buss (då har ein i alle fall vist at ein forstår skjema over mikroarkitektur). F eks til løysing:

Må dele over fleire mikroinstruksjonar sidan kunn ein operand kan på B-buss
ALU: A, C: H, B: LV (mellomlagre LV i H
ALU: A + B, C: H, B: MDR (H = LV + MDR)
ALU: A + B, C: PC, B: PC (PC = H + PC)
Kan løysast på andre måtar også (rekkefølgje på addisjon og mellomlagring), men må mellomlagre for å kunne summere alle operandar.
LV blir kunn lest, MAR (MDR viss nokon velg å bytte her og så ok (grunna typo, godkjenner alt som er rett) kunn lest. I denne instruksjonener det PC (og H) som skal endrast.

c. IJVM-registrene i figur 6 er satt til følgende verdiar:

”SP”: hex(0001),
”LV”: hex(0002),
”CPP”: hex(0003),
”TOS”: hex(0004),
”OPC”: hex(0005),
”H”: hex(FF0A).

Hva er innholdet i H-registeret og OPC-registeret etter at følgende mikroinstruksjoner har blitt utført? Oppgi svaret i hex-format.

1: ALU: 010100, C: 100000000, Mem: 000 og B: 0110
2: ALU: 011100, C: 101000000, Mem: 000 og B: 0111
3: ALU: 001100, C: 001000000, Mem: 000 og B: 1000

Answer: 1:

ALU: 010100 (B) C: 100000000 (H) Mem: 000 (ingen mem opprasjon) B: 0110 (6 CPP) (H = CPP)

2
ALU: 011100 (OR) C: 101000000 (H og TOS) Mem: 000 (ingen mem opprasjon) B: 0111 (7 TOS) (H = H OR TOS)

3
ALU: 001100 (AND) C: 001000000 (TOS) Mem: 000 (ingen mem opprasjon) B: 1000 (8 OPC) H (uendra), TOS = H AND OPC
(H = 0007, OPC = 0005). Her blir det kunn skrevet til H-registeret (og TOS). OPC blir kunn lest (kopiert) slik at det endrar seg ikkje.

OPPGAVE 4: INSTRUKSJONSSETT ARKITEKTUR (ISA)(25 % ((5 % PÅ A, B,C OG 10 % PÅ D))

En avært enkel prosessor har en load, en store, 8 ALU-instruksjoner og noen spesialinstruksjoner, inkludert NOP-instruksjonen og to flytkontrollinstruksjoner (flow control instructions). Instruksjonsformatet for instruksjonene er vist i figur 5. Alle register og bussar er 32-bit. Det er 16 generelle register tilgjengelig. Prosessoren har en Harvard architecture.

Instructions set:

LOAD: Load data from memory.

load Ri, Rj Load register Ri from memory location in Rj.

STORE: Store data in memory.

store Ri, Rj Store register Ri in memory location in Rj.

ALU: Data manipulation, register–register operations.

ADD Ri, Rj, Rk ADD, $Ri = Rj + Rk$. Set Z-flag if result =0.
NAND Ri, Rj, Rk Bitwise NAND, $Ri = \overline{Rj \cdot Rk}$. Set Z-flag if result =0.
OR Ri, Rj, Rk Bitwice OR, $Ri = Rj + Rk$. Set Z-flag if result =0.
INV Ri, Rj Bitwice invert, $Ri = \overline{Rj}$. Set Z-flag if result =0.
INC Ri, Rj Increment, $Ri = Rj + 1$. Set Z-flag if result =0.
DEC Ri, Rj Decrement, $Ri = Rj - 1$. Set Z-flag if result =0.
MUL Ri, Rj, Rk Multiplication, $Ri = Rj * Rk$. Set Z-flag if result =0.
CMP, Ri, Rj Compare, Set Z-flag if $Ri = Rj$

Special: Misc.

CP Ri, Rj Copy, $Ri < -Rj$ (copy Rj into Ri)
NOP Waste of time, 1 clk cycle.
MOVC Ri, constant Put a constant in register $Ri = C$.

Flow control: Branch.

BZ, Ri Conditional branch on zero, $PC = Ri$.
RT Return, return from branch.

Ri, Rj and Rk: Any user register.
DC: Don't care.

a. Hva slags type instruksjoner er:
i) BZ?
ii) RT?

Answer: i) BZ er ein 1 adresse instruksjon. 1 operand (og det er som angitt ein flytkontroll instruksjon.).
ii) RT har ingen operandar og er ein 0 adresse instruksjon (også flytkontroll som angitt)

b. Hva slags type adresseringsmodi (addressing modes) bruker:
i) MOVC?
ii) CP?

Answer: MOVC: Imidiate. operand verdi er gitt i operand feltet i instruksjonen. operanden vil då befinne seg i programminne.
CP: nyttar registeradressering, kvar operand er eit register (register-register).

c. Kan LOAD- og STORE-instruksjonene adressere hele adresserommet (32-bit adressebusser)?

Answer: Ja, begge brukar 32 bits register som kilde for å adressere minne (til 32 bit minnebussar). Ein kan då peike på alle adresser i adresserommet på 32 bit.

d. Følgende psaudokode er en del av et større program. Svar på spørsmålene ut i fra tilgjengelig innformasjon.

- MOVC R1, 0x00 55;
- MOVC R2, 0x00 AA;
- STORE R1, R2;
- ADD R1, R1, R3;
- STORE R1, R3;
- LOAD R1, R2
- INV R3, R3;
- CMP R1, R2;
- BZ R2;

i) Hva slags adresse vil BZ hoppe til hvis betingelsene for et hopp er til stede?
ii) Vil psaudokode føre til at Z-flagget blir aktivert slik at betingelsene for et hopp er til stede? Begrunn svaret.

Answer: i) BZ vil hoppe til 0x00 AA viss betingelsar for hopp er tilstades (innholde i R2 når BZ instruksjonen blir exekvert). Finn det ved å sjå på koden (Koden er del av eit større kodesegment, innhold i R3 påvirkar ikkje spørsmåla):

- MOVC R1, 0x00 55;
  R1 = 0x00 55
- MOVC R2, 0x00 AA;
  R2 = 0x00 AA
- STORE R1, R2;
  Innold R1 (0x00 55) lagra i adresse 0x00 AA (gitt i R2)

- ADD R1, R1, R3;
  R1 = innhold i R1 + innhold i R3 (kva no det er)
- STORE R1, R3;
  Innold R1 lagra i adresse som er gitt i R3
- LOAD R1, R2;
  R1 = innhold i minnelokasjon gitt i R2 (0x00 AA) som er (0x00 55 (R1 = 0x0055))
- INV R3, R3;
  Invertarar R3
- CMP R1, R2;
  samanlignar innhald i R1 og R2 (0x00 55 med 0x00 AA) ikkje likt Z = 0
- BZ R2;
  Z = 0 ikkje noko hopp.

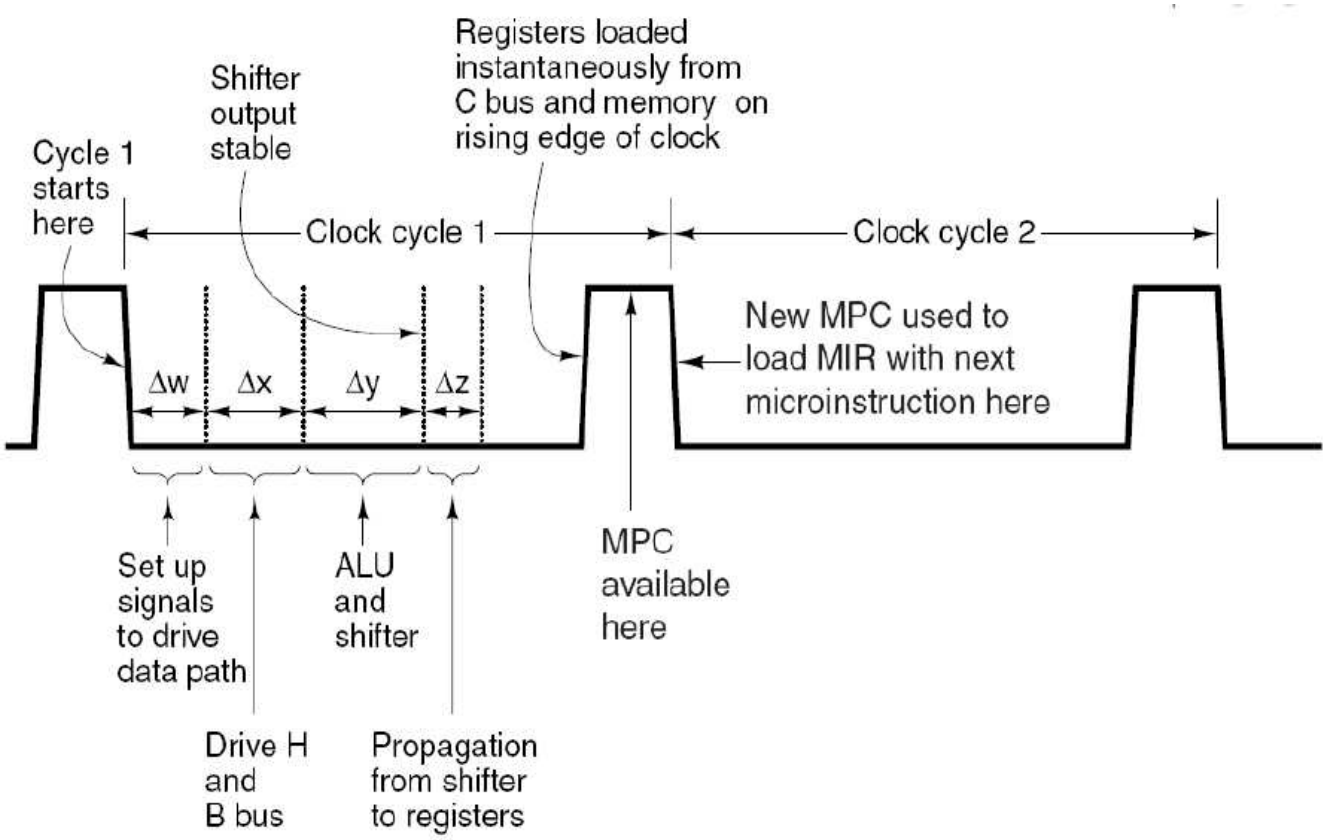
ii) Nei CMP samanlignar R1 som er 0x00 55 og R2 som er 0x00 AA så Z = 0 og programmet vil fortsette med neste instruksjon (ikkje vist).

IJVM appendix

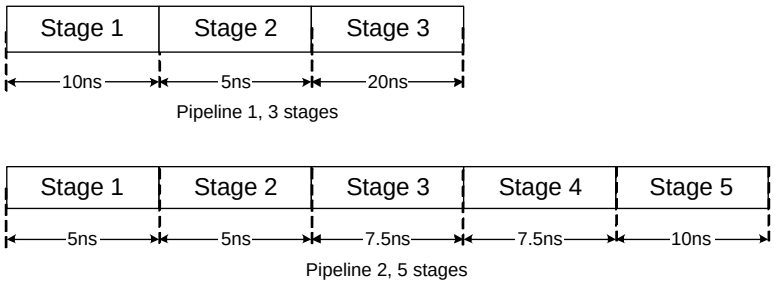
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

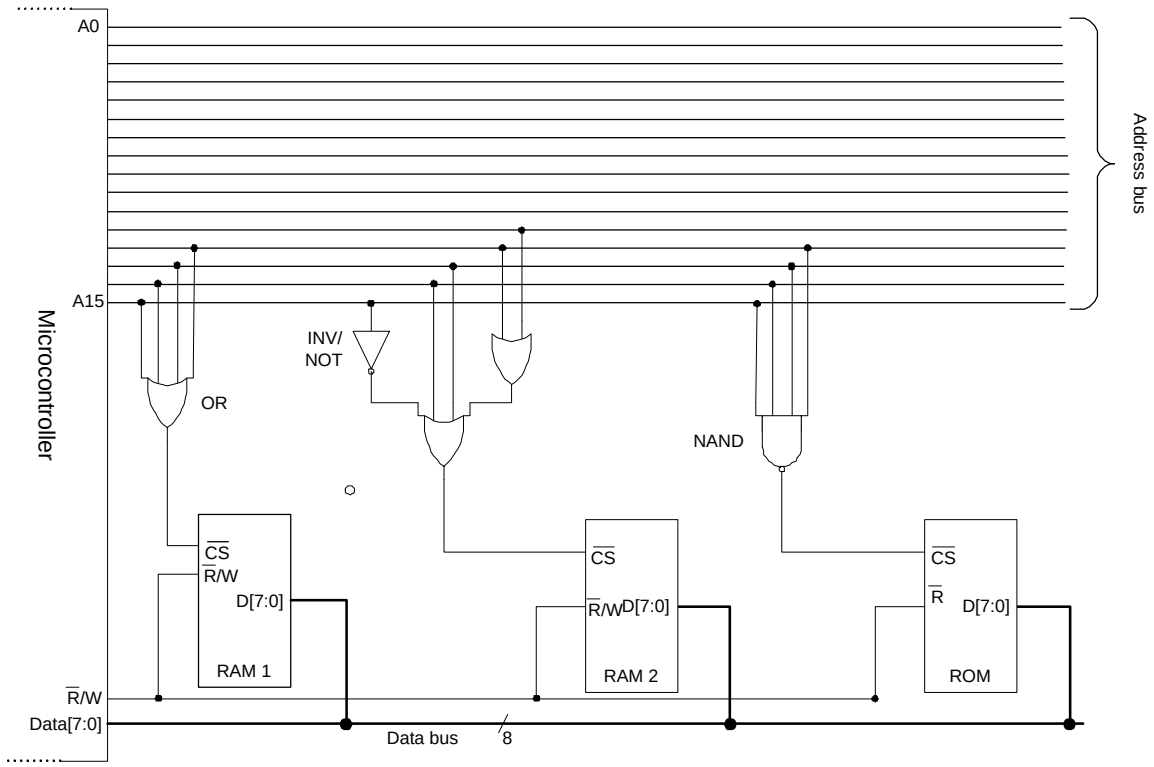
Figur 8: ALU functions (IJVM).



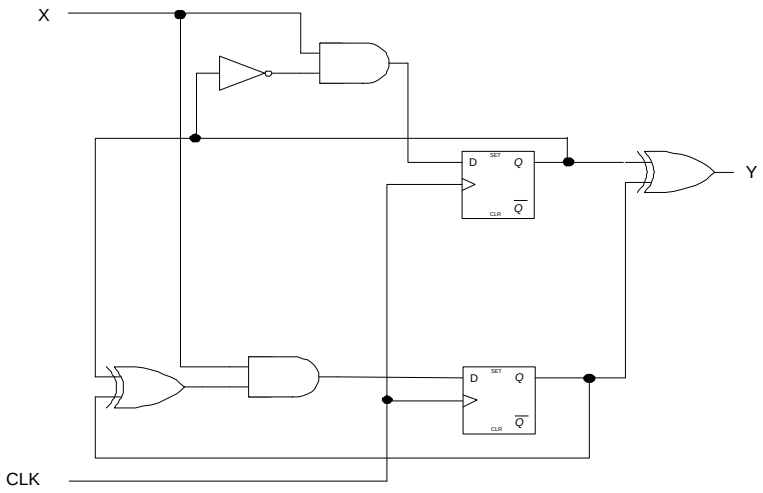
Figur 9: Timing diagram (IJVM).



Figur 10: Samleband for alternativ 2 og 3.



Figur 11: Address decoding.



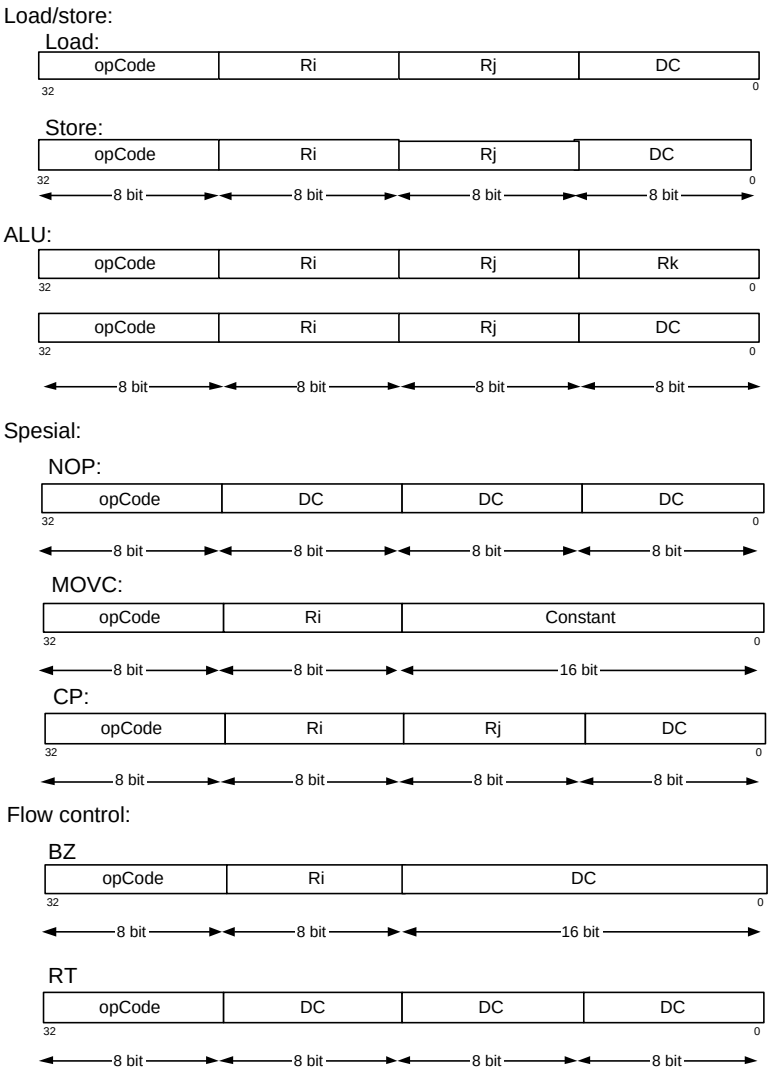
Figur 12: FSM.

Q0 Q1	Q0(nxt) Q1(nxt)	
	X = 0	X = 1
0 0		

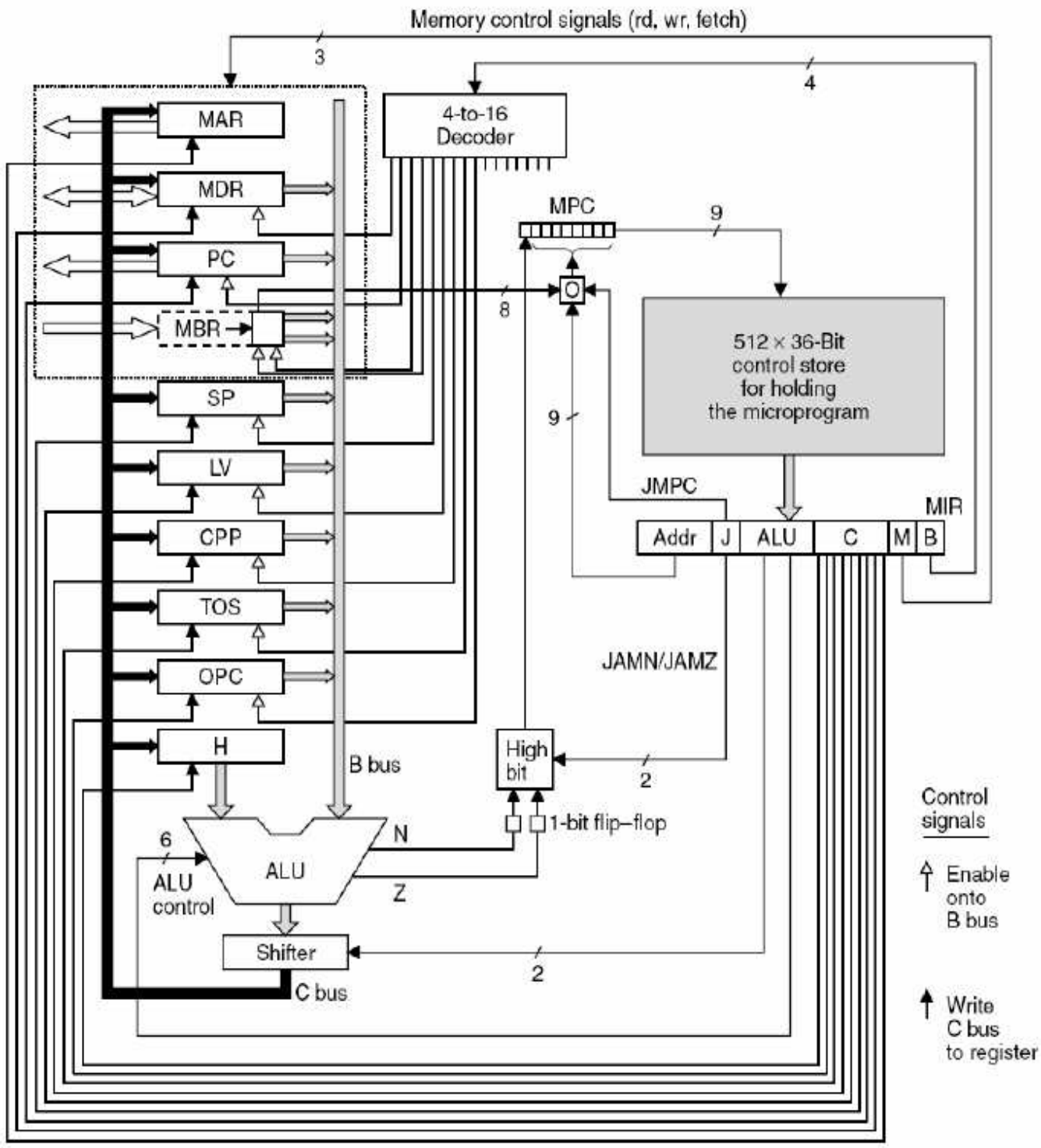
Figur 13: FSM tabell format.

Present state	Next state		Output
Q0 Q1	Q0(nxt) Q1(nxt)		Y
	X = 0	X = 1	
0 0	0 0	1 0	0
0 1	0 0	0 1	1
1 0	0 0	1 1	1
1 1	0 0	0 0	0

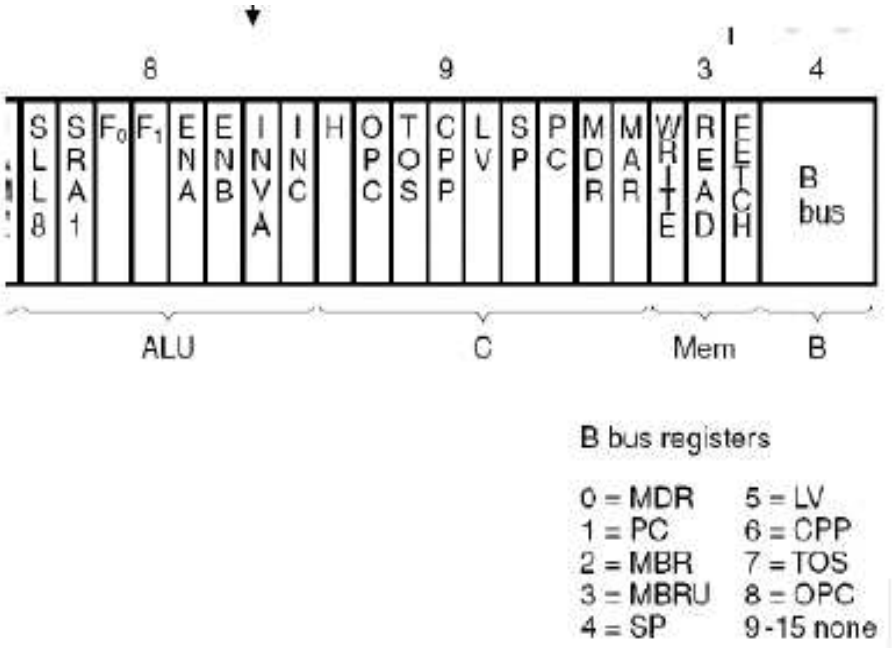
Figur 14: Forslag til utfylt tabell, har også tatt med Y.



Figur 15: Instruction formats.



Figur 16: Block diagram (IJVM).

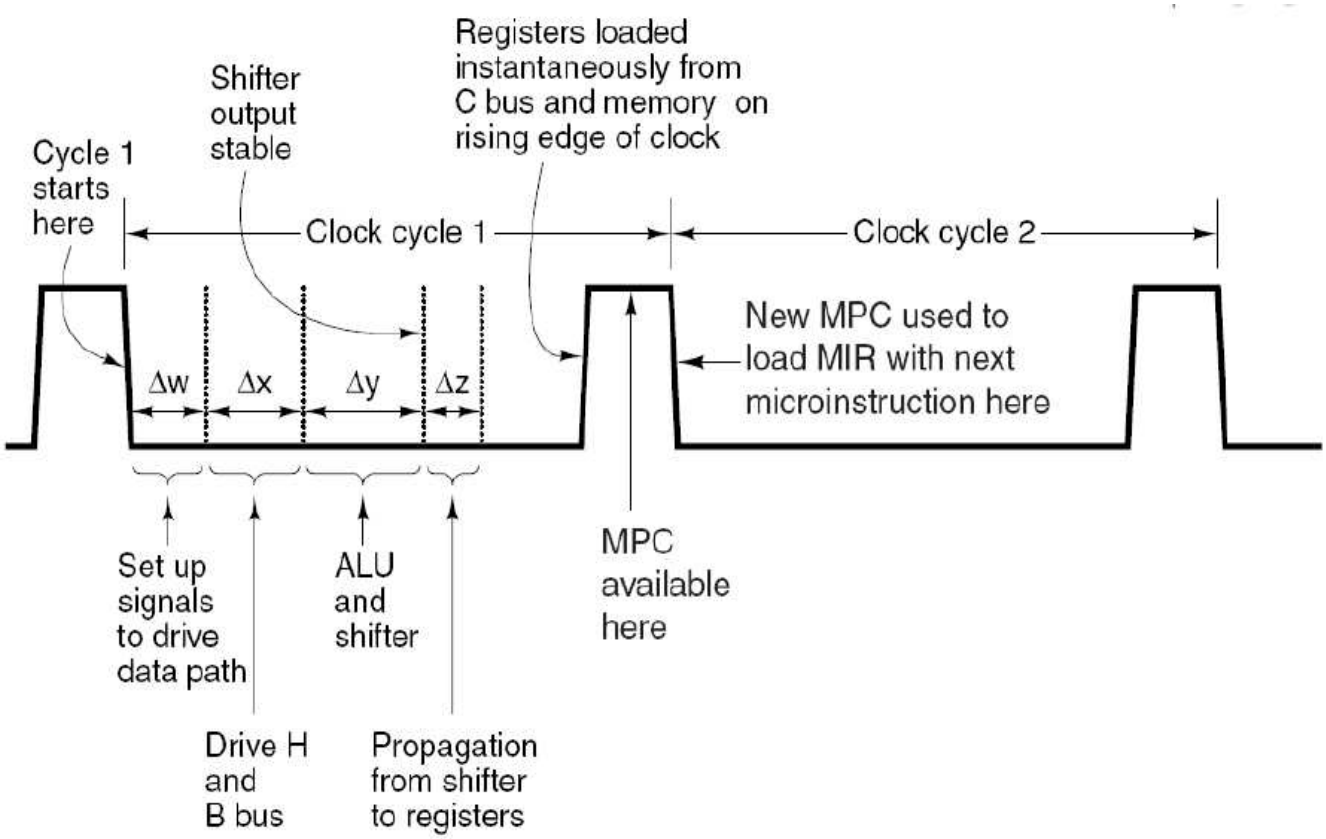


Figur 17: Microinstruction format (IJVM).

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	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
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 18: ALU functions (IJVM).



Figur 19: Timing diagram (IJVM).