



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

TDT4160
DATAMASKINER GRUNNKURS
EKSAMEN

4. DESEMBER, 2014, 09:00–13:00

Kontakt under eksamen:

Gunnar Tufte 73590356/97402478

Tillatte hjelpemidler:

D.

Ingen trykte eller håndskrivne hjelpemiddel er tillete.

Enkel godkjent kalkulator er tillete.

Målform:

Nynorsk

OPPGÅVE 1: OPPSTART, LITT AV KVART (25 %)

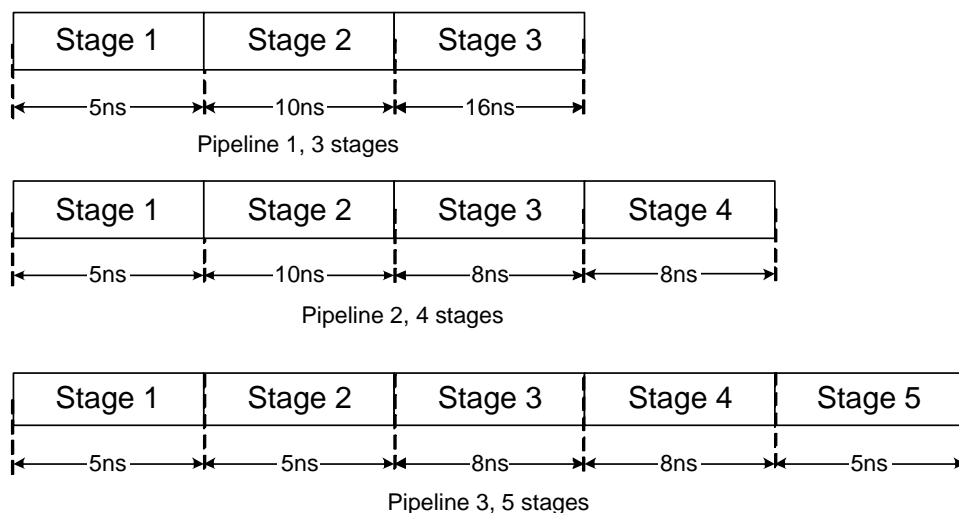
- a. Eit minnesystem med eit nivå hurtigbuffer (cache) der hovudminnet har en aksesstid på $1\mu s$ og hurtigbuffer har ein aksesstid på $0,05\mu s$ med et trefforholdstal (hit ratio) på 80 %.

Du har to valg for å auke ytelsen:

- 1 Endre designet på hurtigbufferet til å ha eit trefforholdstal på 95 %.
- 2 Bytte til eit hovudminne med aksesstid på $0,5\mu s$

Kva alternativ vil gi det raskaste minnesystemet? Kva er ytelseforbedringen?

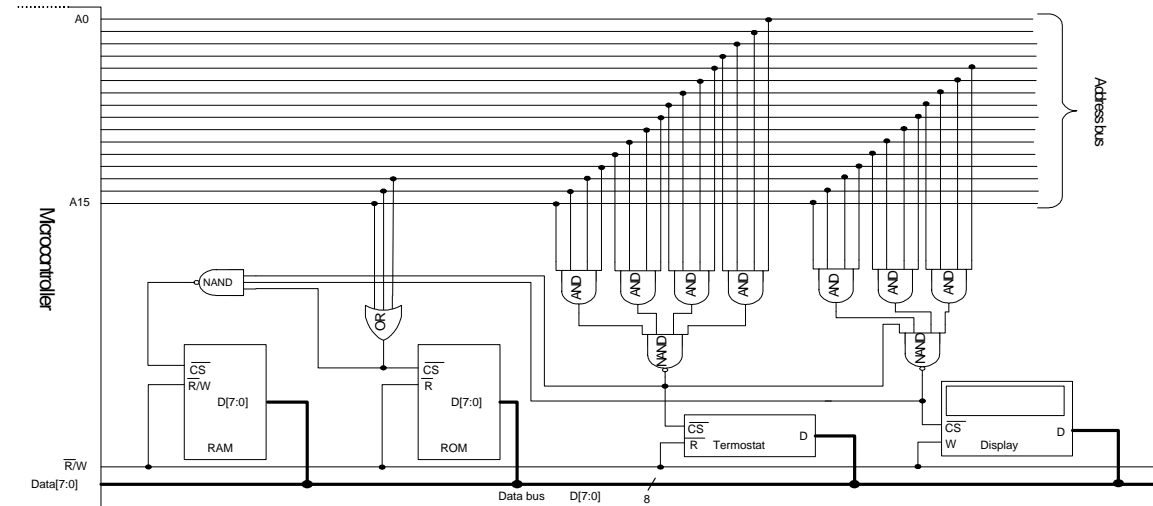
- b. Kvifor blir dynamisk minne som oftast brukt som teknologi i hovudminne?
- c. Forklar kort "Moore's law" (Moore sin lov).
- d. Er felles ISA nødvendig i eit "Heterogeneous" Chip Multiprocessor (CMP) system?
- e. Figur 1 viser tidsforbruket i tre ulike samleband (pipelines). Alle er ulike implementasjonar av den samme ISA. Estimer maksimal klokkefrekvens for dei tre samlebanda. Kva samleband har høgast ILP?



Figur 1: Samlebandtidsforbruk.

OPPGÅVE 2: DIGITALT LOGISK NIVÅ (25 % (15 % PÅ A, 5 % PÅ B OG C))

I figur 2 er det eksterne bussgrensesnittet for ein mikrokontroller vist. Det er brukt ein RAM-brikke for data og ein ROM-brikke for program. Ein termostat overvakar temperaturen. Eit display angir temperatur. Alle einingane nyttar eit aktivt lågt (logisk "0") CS (Chip Select)-signal.



Figur 2: Address decoding.

- a.
 - i) Tegn minnekart for systemet.
 - ii) Kva er det største programmet (i byte) som får plass i ROM.
- b.
 - i) Angi antal adresser som er tilgjengeleg for diplayeiningen.
 - ii) Er det overlapp i minnesystemet? I så fall, korleis påverkar det funksjonaliteten?
- c. Ut i frå den informasjonen du har tilgjengeleg:
 - i) Er det mogleg å utvide systemet med fleire einingar? Forklar.
 - ii) Er påstanden om at systemet er programert til å bruke programstyrt I/O (polling/bussy wait) sansynlig? Forklar.

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

Bruk vedlagte diagram i figur 4, figur 5, figur 6 og figur 7 for IJVM til å løyse oppgåvene.

- a. MBR-registeret er på 8 bit, men MPC-registeret er på 9 bit. Kva type instruksjonar nyttar dette niende bitet? Forklar kort.
- b. Lag mikroinstruksjon(ar) for følgjande IJVM-operasjon: $TOS = TOS + LV + OPC$.
Du trenger ikkje ta omsyn til Addr- og J-feltene. Oppgi korrekt bit-verdi for ALU-, C-, Mem- og B-feltene. – Sjå figur 5.
- c. IJVM-registerna i figur 4 er sett til følgjande verdier:

"SP": hex(AAAA),
"LV": hex(0505),
"CPP": hex(5555),
"TOS": hex(0004),
"OPC": hex(0005),
"H": hex(FF0A).

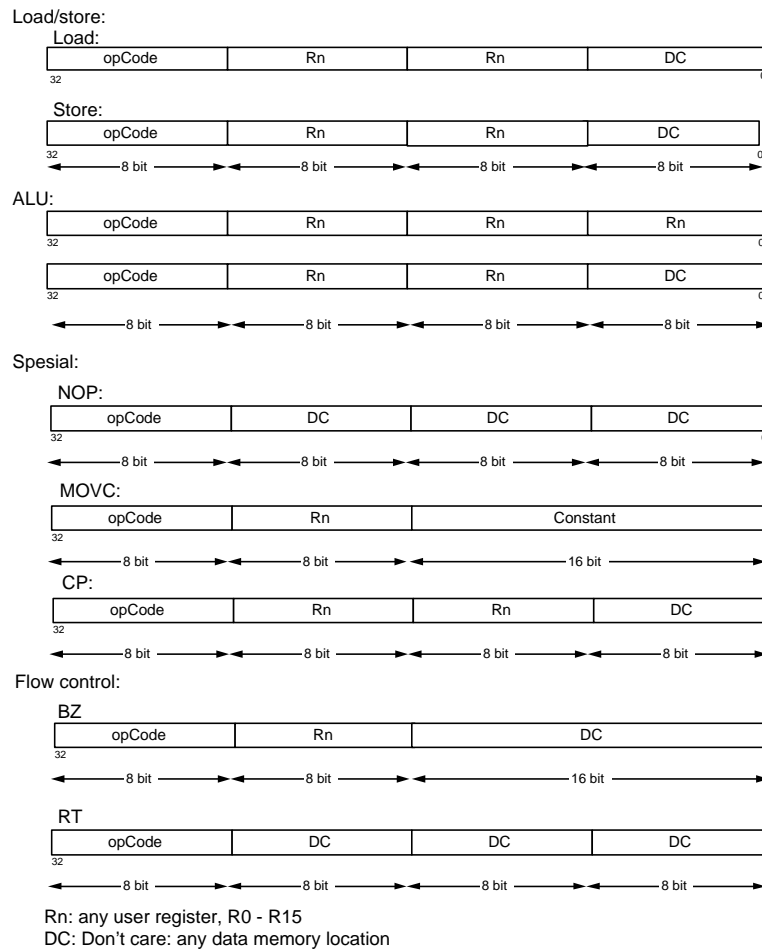
Kva er innhaldet i TOS-registeret etter at dei to følgjande mikroinstruksjonane har blitt utført? Oppgi svaret i hex-format.

1: ALU: 010100, C: 10000000, Mem: 000 og B: 0110
2: ALU: 001100, C: 00100000, Mem: 000 og B: 0100

Kva vil N- og Z-bita være etter utføringa av mikroinstruksjonane?

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

Ein sær enkel prosessor har ein load, ein store, 8 ALU-instruksjonar og nokre spesialinstruksjonar, inkludert NOP-instruksjonen og to flytkontrollinstruksjonar (flow control instructions). Prosessoren har 16 generelle register som er tilgjengeleg. Instruksjonsformatet for instruksjonane er vist i figur 3. Alle register og bussar er 32-bit. Prosessoren har ein Harvard arkitektur.



Figur 3: Instruction formats.

Instructions set:

LOAD: Load data from memory.

load Rn, Rn Load register Rn from memory location in Rn.

STORE: Store data in memory.

store Rn, Rn Store register Rn in memory location in Rn.

ALU: Data manipulation, register–register operations.

ADD Rn, Rn, Rn ADD, $Rn = Rn + Rn$. Set Z-flag if result =0.

NAND Rn, Rn, Rn Bitwise NAND, $Rn = \overline{Rn \cdot Rn}$. Set Z-flag if result =0.

OR Rn, Rn, Rn Bitwise OR, $Rn = Rn \text{ or } Rn$. Set Z-flag if result =0.

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

INC Rn, Rn Increment, $Rn = Rn + 1$. Set Z-flag if result =0.

DEC Rn, Rn Decrement, $Rn = Rn - 1$. Set Z-flag if result =0.

MUL Rn, Rn, Rn Multiplication, $Rn = Rn * Rn$. Set Z-flag if result =0.

CMP, Rn, Rn Compare, Set Z-flag if $Rn = Rn$

Special: Misc.

CP Rn, Rn Copy, $Rn \leftarrow Rn$

NOP Waste of time, 1 clk cycle.

MOVC Rn, constant Put a constant in register $Rn = C$.

Flow control: Branch.

BZ, Rn Conditional branch on zero, $PC = Rn$.

RT Return, return from branch.

Rn: Any user register.

DC: Don't care.

- a. Dette er ein RISC-prosessor. Kvifor?
- b. i) Kva type (instruction format) instruksjon er MOVC?
ii) Kva type adresseringsmodi (addressing modes) brukar MOVC?
iii) Kva type adresseringsmodi (addressing modes) brukar CP?
- c. i) Er det mogleg å utvide med fleire register enn 32 utan å endre instruksjonslengda?
Forklar kort.
ii) Er denne prosessoren generell? Forklar kort.
- d. Følgjande register har gitt innhald:
R1 = 0x0000 AAAA,
R2 = 0x0000 5555,
R3 = 0xAAAA AAAA,
R4 = 0x5555 5555.

Følgjande sudo kode køyrer:

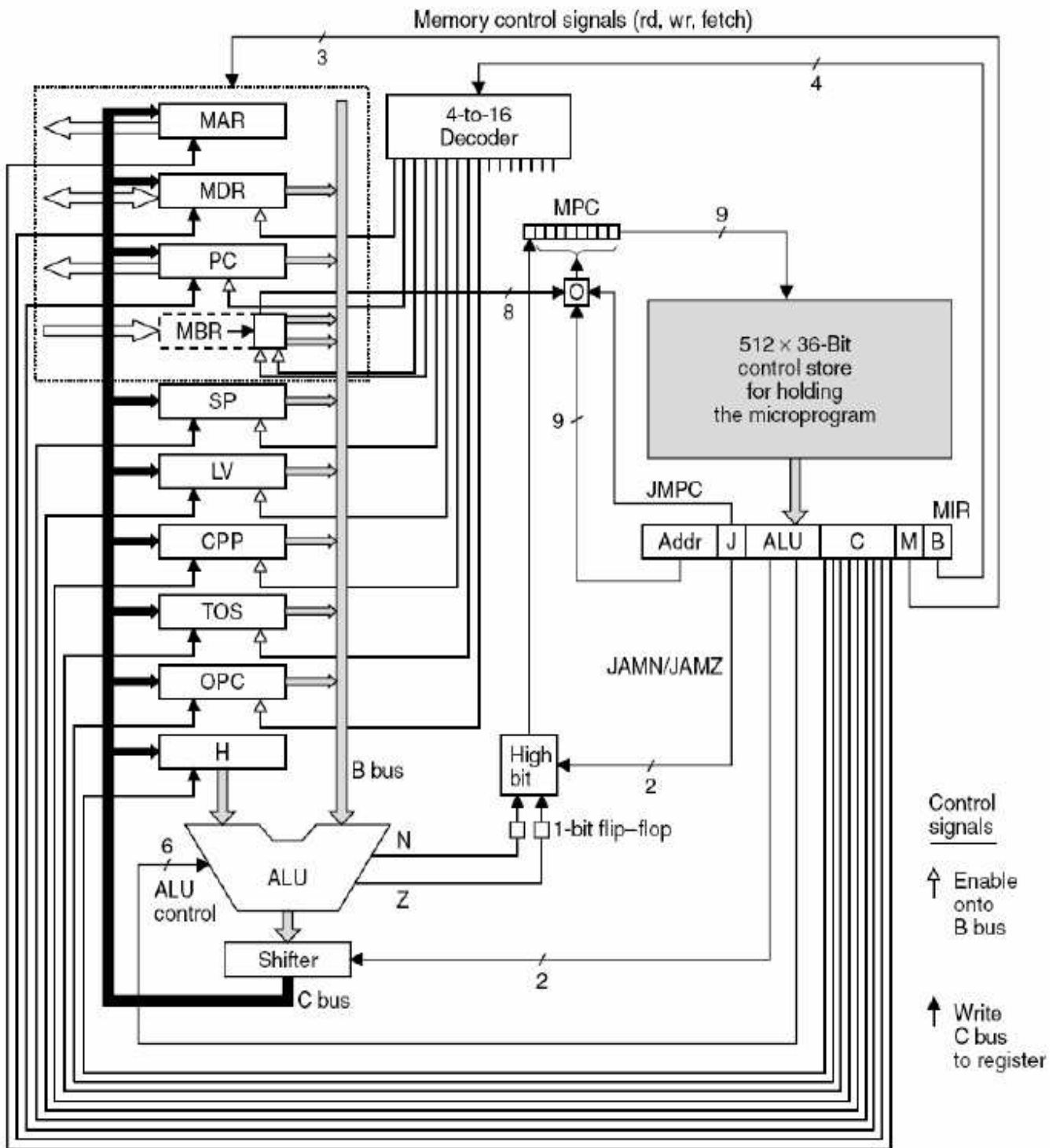
- :
- STORE R1, R3;
- STORE R2, R4;
- INV R3, R3;
- INV R4, R4;
- LOAD R1, R3;
- LOAD R2, R4;

- i) Kva er innhaldet i R1 og R2 etter at koden er køyrt?
ii) I eit forsøk på å optimalisere endrast koden til:

- :
- STORE R1, R3;
- STORE R2, R4;
- CP R1, R2;
- CP R2, R1;

Er resultatet det samme eller ikkje? Kort forklaring.

IJVM appendix



Figur 4: Block diagram (IJVM).

ANSWER KEY FOR THE EXAM

OPPGÅVE 1: OPPSTART, LITT AV KVART (25 %)

- a. Eit minnesystem med eit nivå hurtigbuffer (cache) der hovudminnet har en aksesstid på $1\mu s$ og hurtigbuffer har ein aksesstid på $0,05\mu s$ med et trefforholdstal (hit ratio) på 80 %.

Du har to valg for å auke ytelsen:

- 1 Endre designet på hurtigbufferet til å ha eit trefforholdstal på 95 %.
- 2 Bytte til eit hovudminne med aksesstid på $0,5\mu s$

Kva alternativ vil gi det raskaste minnesystemet? Kva er ytelseforbedringen?

Answer: mean access time orginalt: $c + (1 - h) * m$; $0,05 + (1 - 0,8) * 1 = 0,25\mu s$

For 1: $0,05 + (1 - 0,95) * 1 = 0,1\mu s$

For 2: $0,05 + (1 - 0,8) * 0,5 = 0,15\mu s$

også med $c * h + (1 - h) * m$), men får andre verdier:

Orginal: $0,24\mu s$, 1: $0,0975\mu s$ og 2: $0,1475\mu s$.

Alternativ 1 er raskast med ein forbedring i gjennomsnitteleg aksesstid på $0,15\mu s$ (eller 2,5 x ytelse forbedring). $0,1425\mu s$ viss $c * h + (1 - h) * m$ brukt.

- b. Kvifor blir dynamisk minne som oftast brukt som teknologi i hovudminne?

Answer: Dynamisk:

Ikkje så rask som statisk MEN:

Liten minnecelle (areal)

Lite effektforbruk

+ andre fornuftige grunnar.

- c. Forklar kort "Moore's law" (Moore sin lov).

Answer: Exponensiell vekst av antall gates det er plass til på ein brikke. (ca dobling kvar 18 månad).

- d. Er felles ISA nødvendig i eit "Heterogeneous" Chip Multiprocessor (CMP) system?

Answer: Nei, prosessorar kan være forskjellige. F.eks. Generelle og GPGPU blanda.

- e. Figur 1 viser tidsforbruket i tre ulike samleband (pipelines). Alle er ulike implementasjonar av den samme ISA. Estimer maksimal klokkefrekvens for dei tre samlebanda. Kva samleband har høgast ILP?

Answer: Treigastetrinn bestemmer max klokke.

Pipeline klokkehastighet: $1/t$. 3 har høgast ILP (5 mot 4 og 3 inst. inne på brikke).

OPPGÅVE 2: DIGITALT LOGISK NIVÅ (25 % (15 % PÅ A, 5 % PÅ B OG C))

I figur 2 er det eksterne bussgrensesnittet for ein mikrokontroller vist. Det er brukt ein RAM-brikke for data og ein ROM-brikke for program. Ein termostat overvakar temperaturen. Eit display angir temperatur. Alle einingane nyttar eit aktivt lågt (logisk "0") CS (Chip Select)-signal.

- a.
 - i) Tegn minnekart for systemet.
 - ii) Kva er det største programmet (i byte) som får plass i ROM.

Answer: i) Minnekart med adresser trengs.

ROM:0000 - 1FFF,

RAM: 2000 - FFEF,

Termostat: FFFF,

Display: FFF0 - FFFE

ii) hex: 2000 (des: 8192) bytes.

- b.
 - i) Angi antal adresser som er tilgjengeleg for diplayeiningen.
 - ii) Er det overlapp i minnesystemet? I så fall, korleis påverkar det funksjonaliteten?

Answer: i) det er 15 adresser avsatt til displayet.

ii) Det er ingen overlapp.

- c. Ut i frå den informasjonen du har tilgjengeleg:

- i) Er det mogleg å utvide systemet med fleire einingar? Forklar.
- ii) Er påstanden om at systemet er programert til å bruke programstyrt I/O (polling/bussy wait) sansynlig? Forklar.

Answer: i) Nei. Alle minneadresser er brukt.

ii) Det er sansynlig sidan det ikkje er nokon IRQ-linje. Einaste måten å få oppdatert temperaturdata på er å lese med jevne mellomrom.

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

Bruk vedlagte diagram i figur 4, figur 5, figur 6 og figur 7 for IJVM til å løyse oppgåvene.

- a. MBR-registeret er på 8 bit, men MPC-registeret er på 9 bit. Kva type instruksjonar nyttar dette niende bitet? Forklar kort.

Answer: Betingehopp instruksjonar utnyttar bit 9. Brukar det niende bite til å kunne velge to mikroprogram stiar. Ved hopp bit 9 = 1. Då vil mikroinstruksjonane ligge i adr 1xxxxxxx i control store. Viss betingelse for hopp ikkje er oppfylgt vil bit 9 vere 0; og neste microinstruksjon vil ligge i 0xxxxxxx i control store. N og Z status frå ALU er tilgjengeleg som betingelsar.

- b. Lag mikroinstruksjon(ar) for følgjande IJVM-operasjon: $TOS = TOS + LV + OPC$.

Du trenger ikkje ta omsyn til Addr- og J-feltene. Oppgi korrekt bit-verdi for ALU-, C-, Mem- og B-feltene. – Sjå figur 5.

Answer: 1: ALU: B, C: h, B: TOS. i bit: ALU: 010100, C: 100000000, MEM: 0, B: 7 (0111)

2: ALU: A + B, C: h, B: LV. i bit: ALU: 111100, C: 100000000, MEM: 0, B: 5 (0101)

3: ALU: A + B, C: TOS, B: OPC. i bit: ALU: 111100, C: 001000000, MEM: 0, B: 8 (1000)

Og alle andre omstokkingar av rekkefølge av addisjon av register som gir rett svar resultat i TOS.

- c. IJVM-registerna i figur 4 er sett til følgjande verdier:

"SP": hex(AAAA),

"LV": hex(0505),

"CPP": hex(5555),

"TOS": hex(0004),

"OPC": hex(0005),

"H": hex(FF0A).

Kva er innhaldet i TOS-registeret etter at dei to følgjande mikroinstruksjonane har blitt utført? Oppgi svaret i hex-format.

1: ALU: 010100, C: 100000000, Mem: 000 og B: 0110

2: ALU: 001100, C: 001000000, Mem: 000 og B: 0100

Kva vil N- og Z-bita være etter utføringa av mikroinstruksjonane?

Answer: 1: ALU: 010100 (B) C: 100000000 (H) Mem: 000 (ingen mem opprasjon) B: 0110 (6 CPP)

2: ALU: 111101 (A and B) C: 001000000 (TOS) Mem: 000 (ingen mem opprasjon) B: 0100 (4 SP)

TOS = 0000, gjer funksjonen: CPP AND SP (0x5555 AND 0xAAAA)

Z = 1 og N = 0. Resultatet var 0 så då vil Z bli aktivert.

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

Ein sær enkel prosessor har ein load, ein store, 8 ALU-instruksjonar og nokre spesialinstruksjonar, inkludert NOP-instruksjonen og to flytkontrollinstruksjonar (flow control instructions). Prosessoren har 16 generelle register som er tilgjengeleg. Instruksjonsformatet for instruksjonane er vist i figur 3. Alle register og bussar er 32-bit. Prosessoren har ein Harvard arkitektur.

Instructions set:

LOAD: Load data from memory.

load Rn, Rn Load register Rn from memory location in Rn.

STORE: Store data in memory.

store Rn, Rn Store register Rn in memory location in Rn.

ALU: Data manipulation, register–register operations.

ADD Rn, Rn, Rn ADD, $Rn = Rn + Rn$. Set Z-flag if result =0.

NAND Rn, Rn, Rn Bitwise NAND, $Rn = \overline{Rn \cdot Rn}$. Set Z-flag if result =0.

OR Rn, Rn, Rn Bitwise OR, $Rn = Rn \text{ or } Rn$. Set Z-flag if result =0.

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

INC Rn, Rn Increment, $Rn = Rn + 1$. Set Z-flag if result =0.

DEC Rn, Rn Decrement, $Rn = Rn - 1$. Set Z-flag if result =0.

MUL Rn, Rn, Rn Multiplication, $Rn = Rn * Rn$. Set Z-flag if result =0.

CMP, Rn, Rn Compare, Set Z-flag if $Rn = Rn$

Special: Misc.

CP Rn, Rn Copy, $Rn \leftarrow Rn$

NOP Waste of time, 1 clk cycle.

MOVC Rn, constant Put a constant in register $Rn = C$.

Flow control: Branch.

BZ, Rn Conditional branch on zero, $PC = Rn$.

RT Return, return from branch.

Rn: Any user register.

DC: Don't care.

a. Dette er ein RISC-prosessor. Kvifor?

Answer: Enkle instruksjonar. Like lange instruksjonar, Load/store arkitektur, mange generelle register.

- b.
- i) Kva type (instruction format) instruksjon er MOVCC?
 - ii) Kva type adresseringsmodi (addressing modes) brukar MOVCC?
 - iii) Kva type adresseringsmodi (addressing modes) brukar CP?

Answer: i) to-adresse instruksjon.
ii) immediate addressing.
iii) Register addressing

- c.
- i) Er det mogleg å utvide med fleire register enn 32 utan å endre instruksjonslengda? Forklar kort.
 - ii) Er denne prosessoren generell? Forklar kort.

Answer: i) Ja kan utvide med fleire register det er satt av 8 bit til register adressering. 16 register brukar 4 bit til adressering, 32 register 5 bit. Det er mulig å adressere totalt 256 register med 8 bit utan å utvide instruksjonslengda.
ii) Ja, Branch, I/O og ALU-funksjonar.

d. Følgjande register har gitt innhald:

R1 = 0x0000 AAAA,
R2 = 0x0000 5555,
R3 = 0xAAAA AAAA,
R4 = 0x5555 5555.

Følgjande sudo kode køyrer:

- :
- STORE R1, R3;
- STORE R2, R4;
- INV R3, R3;
- INV R4, R4;
- LOAD R1, R3;
- LOAD R2, R4;

- i) Kva er innhaldet i R1 og R2 etter at koden er køyrt?
ii) I eit forsøk på å optimalisere endrast koden til:

- :
- STORE R1, R3;
- STORE R2, R4;
- CP R1, R2;
- CP R2, R1;

Er resultatet det samme eller ikkje? Kort forklaring.

Answer: i) Bytter om register innhold ved å å bytte peikarar ved invertering. $R1 = 0x0000$ 5555 og $R2 = 0x0000$ AAAA.

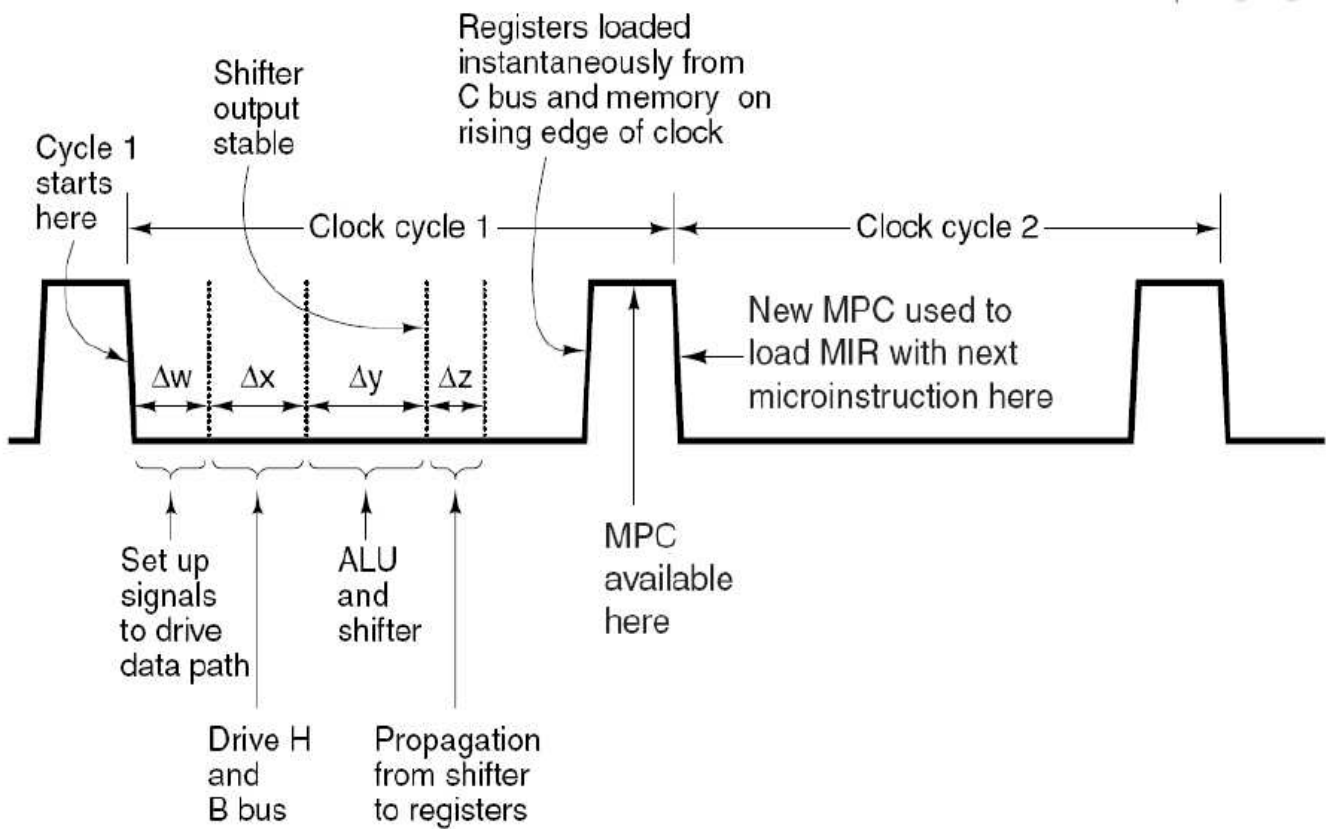
ii) Nei, her blir R2 kopiert til R1. ($R1 = (R2) = 0x0000$ 5555), men så blir verdien i R2 oppdatert med ny R1 verdi, altså $0x0000$ 5555. For at dette skulle virke kunne ein foreksempel mellomagre R1 i eit anna ledigt register.

IJVM appendix

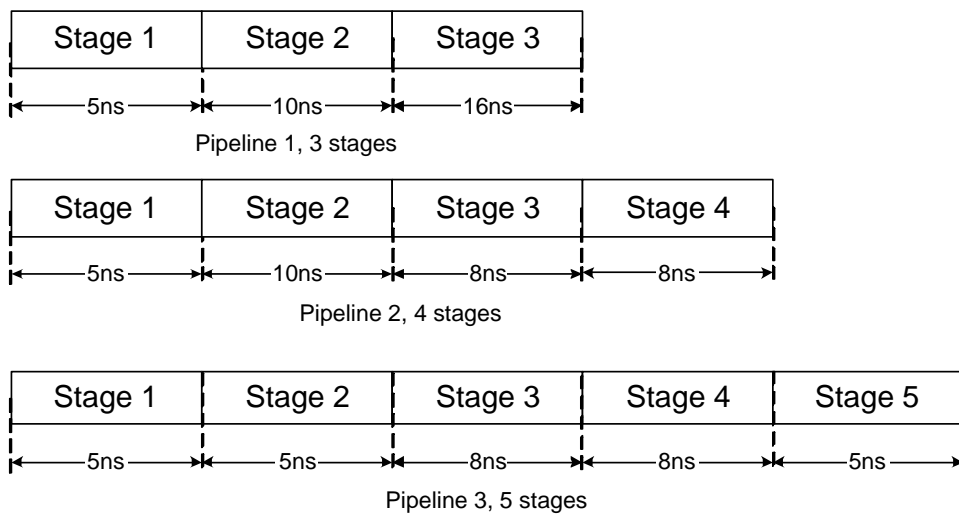
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	\bar{A}
1	0	1	1	0	0	\bar{B}
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B - A
1	1	0	1	1	0	B - 1
1	1	1	0	1	1	-A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0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

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

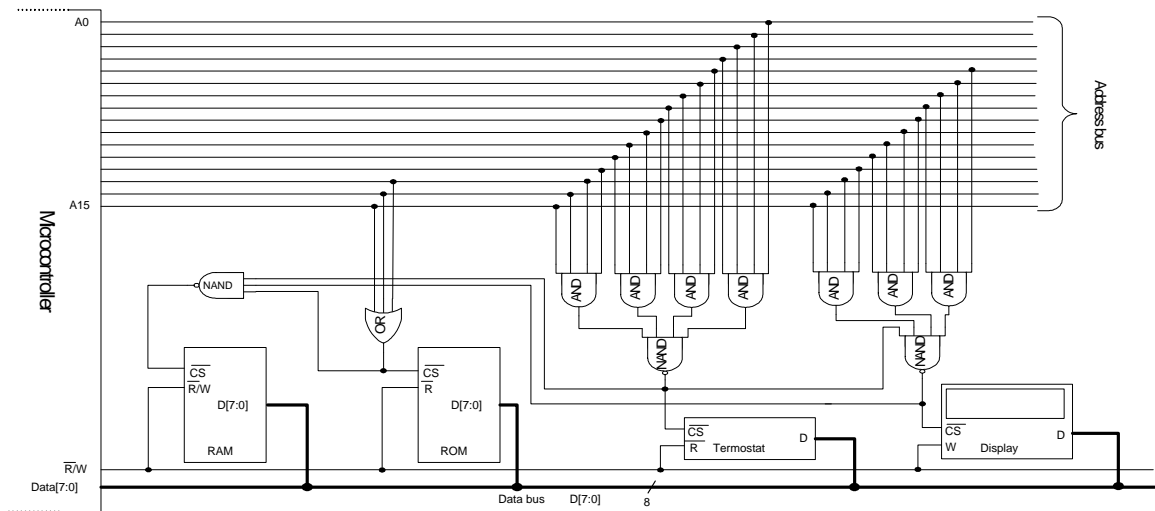
Figur 6: ALU functions (IJVM).



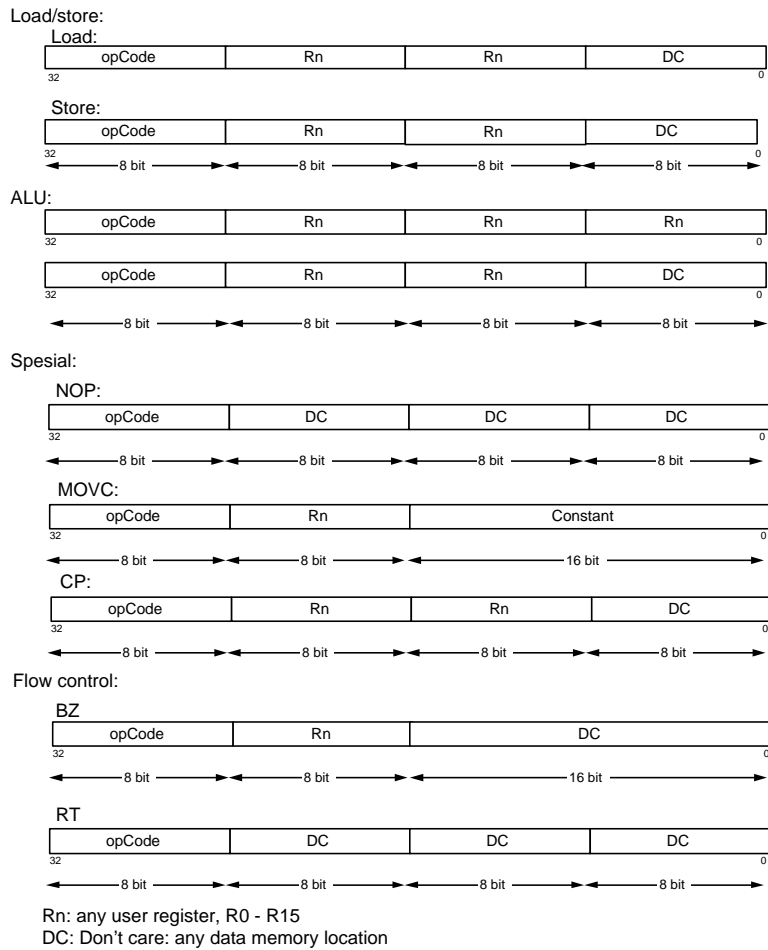
Figur 7: Timing diagram (IJVM).



Figur 8: Samlebandtidsforbruk.



Figur 9: Address decoding.



Figur 10: Instruction formats.

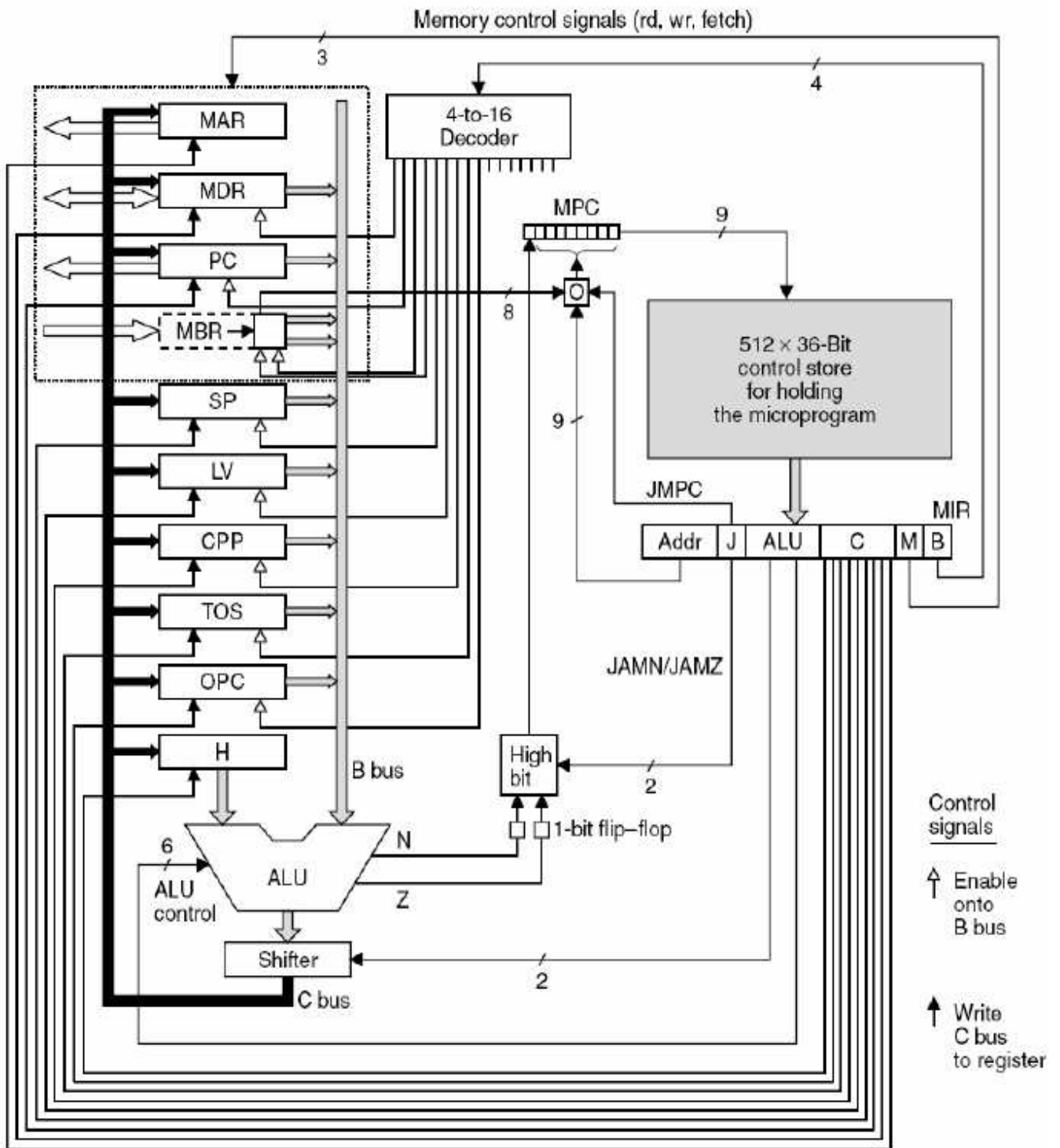
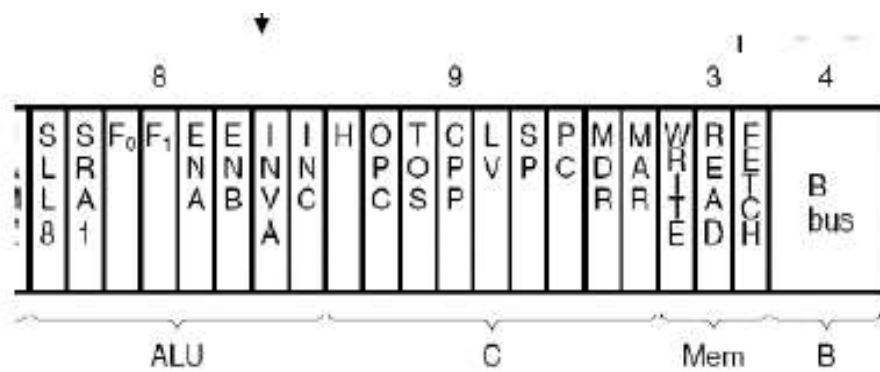


Figure 11: Block diagram (IJVM).



B bus registers

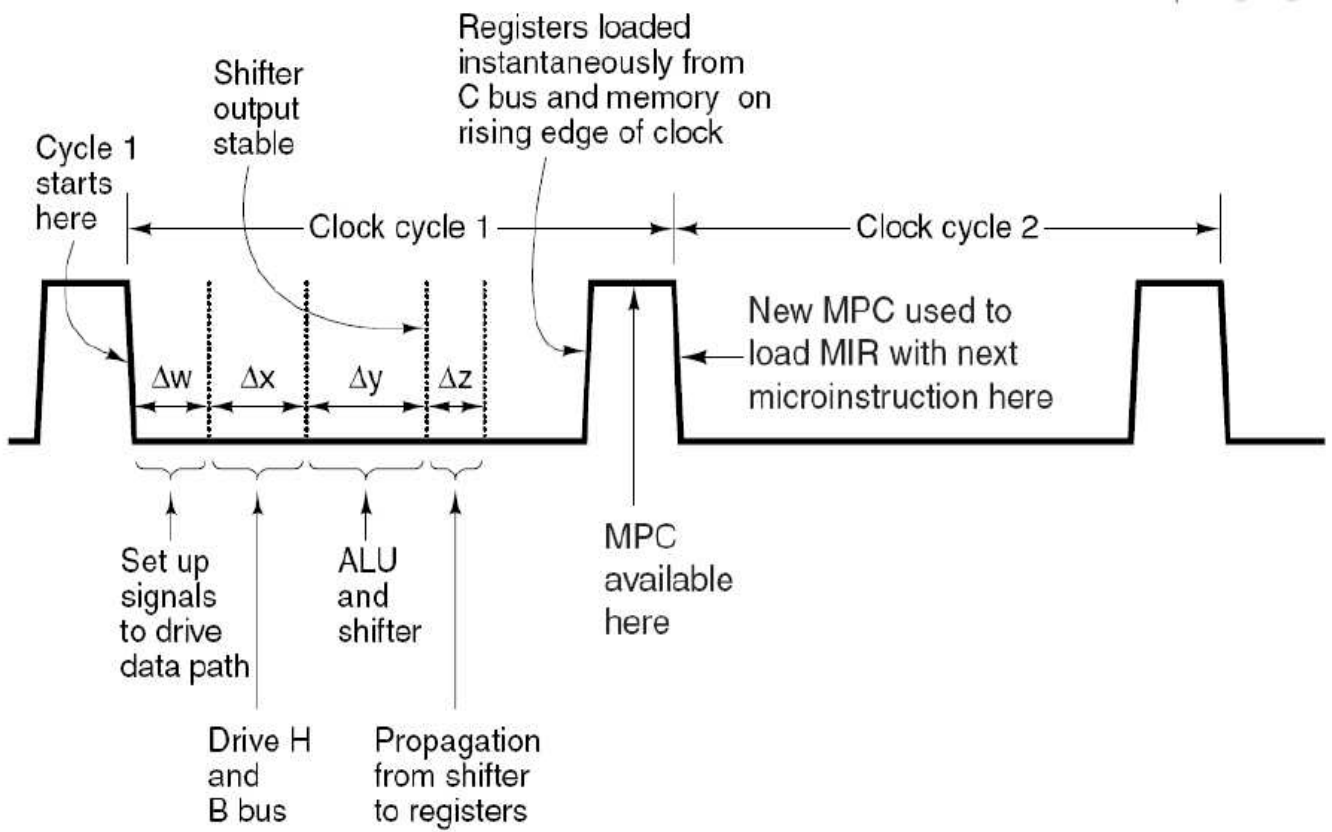
- | | |
|----------|-----------|
| 0 = MDR | 5 = LV |
| 1 = PC | 6 = CPP |
| 2 = MBR | 7 = TOS |
| 3 = MBRU | 8 = OPC |
| 4 = SP | 9-15 none |

Figur 12: 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	\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 13: ALU functions (IJVM).



Figur 14: Timing diagram (IJVM).