NTNU
Kunnskap for en bedre verden

Institutt for datateknikk og informasjonsvitenskap

Eksamensoppgave i TDT4160 datamaskiner og digitalteknikk

Faglig kontakt under eksamen: Gunnar Tufte

Tlf.: 97402478

Eksamensdato: 11. desember 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): 9

Antall sider vedlegg: 3

Informasjon om trykking av eksamensoppgave

Originalen er:

1-sidig x **2-sidig ☐**

sort/hvit x **farger ☐**

skal ha flervalgskjema ☐

Kontrollert av:

Dato Sign

Oppgave 1 Oppstart, litt av hvert (20 % (a: 2,5 %, b: 7,5 %, c:.5 %, d: 2,5 og e: 2,5 %)

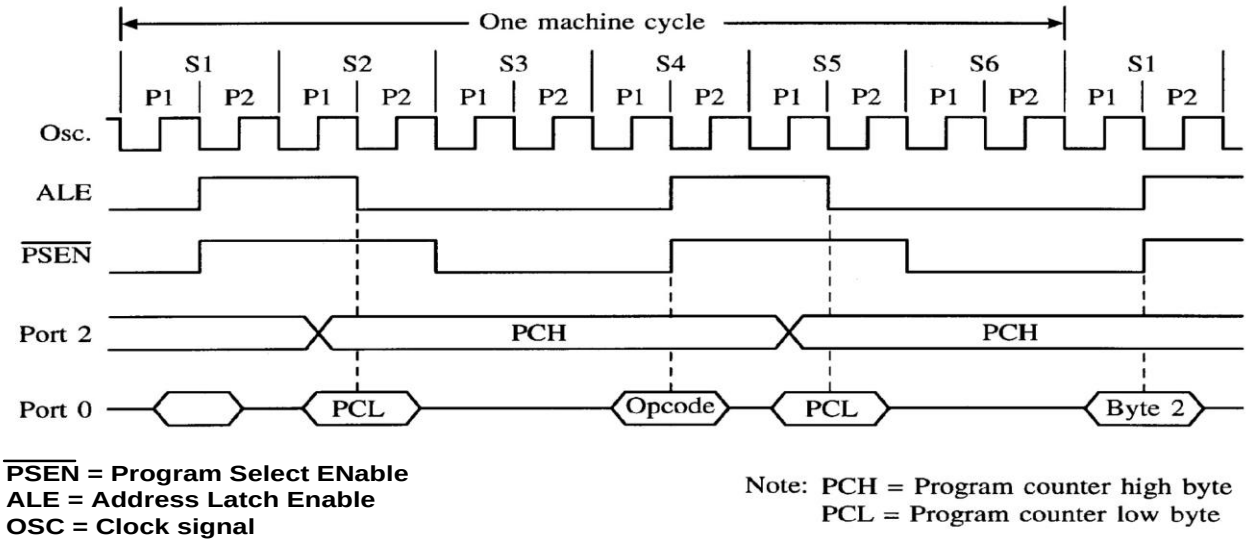
a)

Begrepet «stored program computer» ble innført med von Neumann-arkitekturen. Kva ligger i dette begrepet?

b)

Figur 1 viser en Mikrokontroller sitt grensesnitt mot eksternt minne. Bussignalene i figuren angir lesing av programminnet. Port 0 og Port 2 er 8-bit porter. Svar på følgende spørsmål ut fra tilgjengelig informasjon:

i) Er dette en synkron eller asynkron bussoverføring? Forklar kort.
ii) Hva er størrelsen (maks adresserom) på programminnet? Forklar kort.
iii) Hva er bitbredden på programminnet? Forklar kort.



Figur 1 Bussgrensesnitt.

c)

i) Hva vil det si at en prosessor er «superscalar»?
ii) Kan en innføre superscalaritet uten å påvirke ISA?

d)

Hva kjennetegner en prosessor med Harvard-arkitektur? Forklar kort.

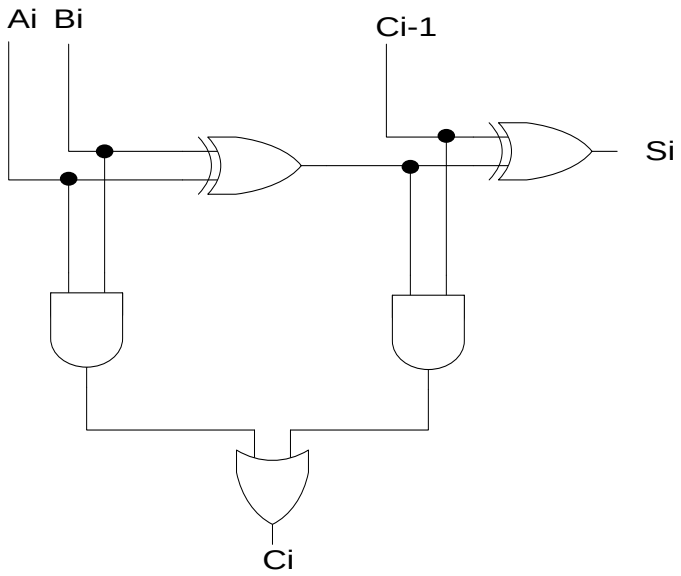
e)

«Latency hiding» er et viktig konsept i moderne datamaskinarkitektur. Er cache en del av dette konseptet? Forklar kort hvorfor, eventuelt hvorfor ikke.

Oppgave 2 Digitalt Logisk Nivå (20 % (a: 4 %, b: 7 %, c: 7 % og 2 % på d))

a)

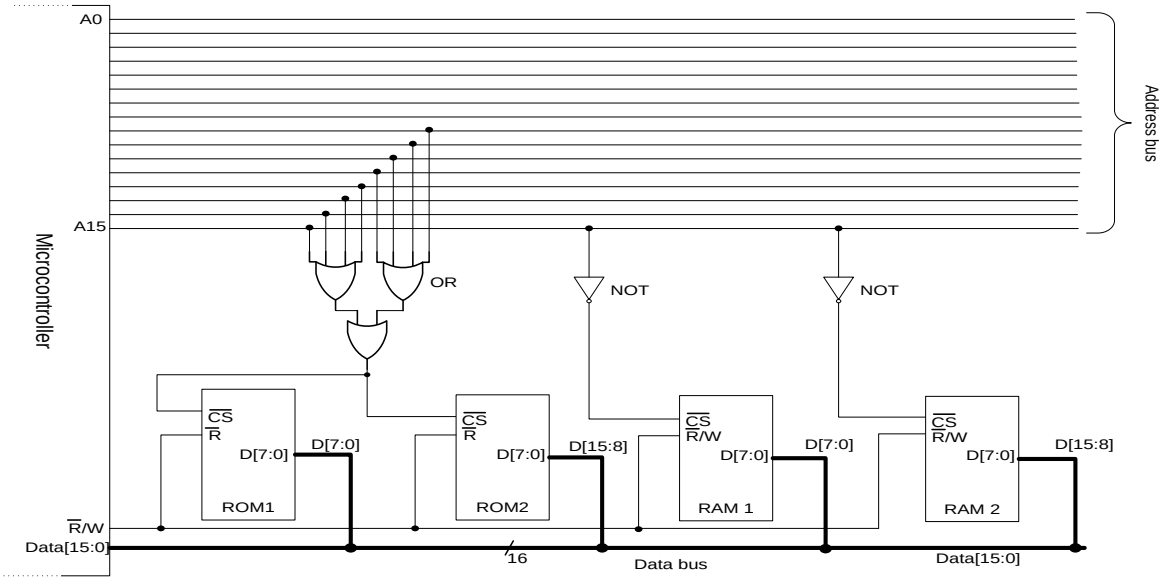
Figur 2 viser logikken til en av funksjonene i en 1 bit ALU. Hvilken funksjon?



Figur 2 Logikk for 1-bit ALU-funksjon.

b)

I et innvevd system (embedded system) brukes en 16-bit mikrokontroller. Figur 3 viser det eksterne bussgrensesnittet med adressedekodingslogikk for mikrokontrolleren. Det er to ROM-brikker for program og to RAM-brikker. Alle enhetene bruker et aktivt lavt (logisk "0") CS (Chip Select)-signal.

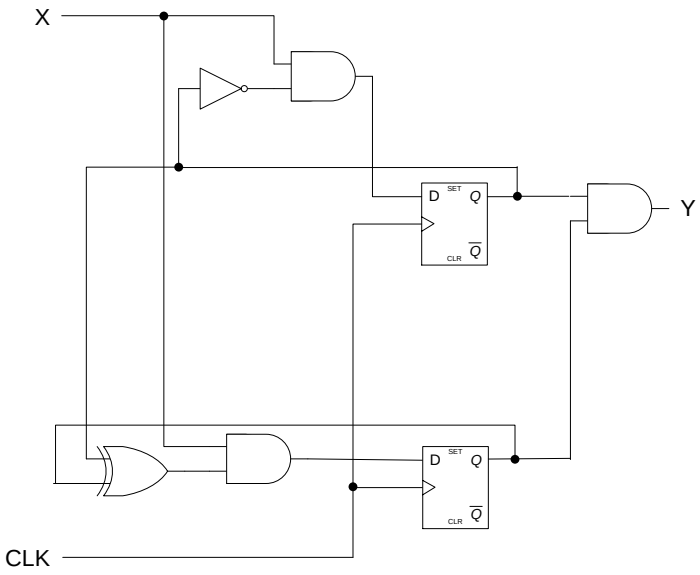


Figur 3 Address decoding.

i) Finn adresseområde for RAM og ROM. Eventuelt ledig minneområde og overlapp.

ii) Tegn minnekart ut fra adresseområde

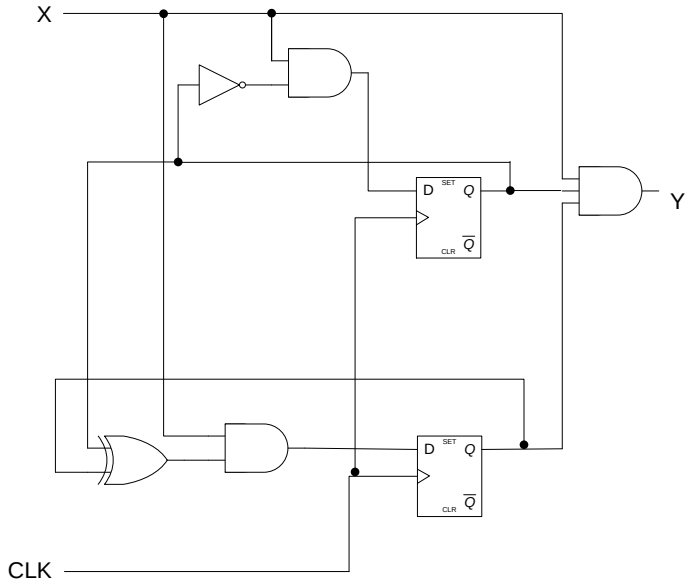
c)
Figur 4 viser en FSM (Finite State Machine). Datavippe med D0 og Q0 er øverst. Datavippe med D1 og Q1 nederst.



Figur 4 Finite State Machine (FSM) 1.

Finn de logiske uttrykkene for D0 og D1 (excitation equation). Angi neste tilstands uttrykk (next state equations) og lag transisjonstabell (next-state-table) for kretsen, som viser oppførselen til denne FSM-en og utgangssignalet Y.

d)
Tilstandsmaskinen i figur 4 blir endret til vist FSM i figur 5. Hvordan påvirker endringen oppførselen til utgangssignalet Y?



Figur 5 Finite State Machine (FSM) 2.

Oppgave 3 Mikroarkitektur og mikroinstruksjoner (20 % (a: 5 %, b: 5 % og 10 % på c))
Bruk vedlagte diagram i figur 9, figur 10, figur 11 og figur12 for IJVM til å løse oppgavene.

a)
Komponenten 4-to-16 Decoder i figur 9 er brukt til å kontrollere hva som skal ligge på B-buss. For C-Bussen er separate bit brukt til å kontrollere hvert register. Hvorfor?

b)
For mikroarkitekturen i Figur 9. Lag mikroinstruksjon(er) som utfører logisk NOR. Data ligg i TOS-registeret og LV-registeret (*TOS NOR LV*). Resultatet skal lagres i H-registeret og MDR-registeret.

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

c)
For mikroarkitekturen i Figur 9. Under utføring av en instruksjon ligger verdien 0xA3 i MPC (MPC peker på adresse 0xA3 i control store).

Innholdet i MDR-registeret er 0xAAAA AAAA.
Innholdet i H-registeret er 0x5555 5555.
Innholdet i LV-registeret er 0xA5A5 0000.

Tabellen under viser to utsnitt av innholdet i control store.
Hva skjer under resten av instruksjonsutføringa? Forklar kort hva hver mikroinstruksjon gjør.
Hvilken verdi vil MDR-registeret ha når MPC får verdien 0x00?

Control store Address	Next_Adr	JMPC	JAMN	JAMZ	SLL8	SRA1	F0	F1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	MDR	MAR	Write	Read	FETCH	B Buss			
0A3	010100100	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0A4	010100101	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1
0A5	010100110	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0A6	000000000	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
													:														
1A5	110100110	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1
1A6	110100111	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1
1A7	000000000	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1A8	110101001	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1
													:														

Oppgave 4 Instruksjonssett arkitektur (ISA) (20 % (a: 2,5 %, b: 2,5 %, c: 10 og 5 % på d))

RaM 007 er en svært enkel prosessor. RaM 007 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)
Bruker RaM 007 fast instruksjonslengde?

b)
Gi et eksempel på en RaM 007 instruksjon som benytter «immediate addressing».

c)
R0 har følgende verdi: 0x0000 FFFF, R8 har følgende verdi: 0xFFFF 0000, R11 har følgende verdi: 0x0000 0000, R12 har følgende verdi: 0x0000 0003 og R13 har følgende verdi 0x0001 0005. I dataminnet ligger følgende data fra adresse 0xFFFF 0000:

Adresse	Data
0xFFFF 0000:	0x00 00 00 01
0xFFFF 0001:	0x00 00 00 02
0xFFFF 0002:	0x00 00 00 03
0xFFFF 0003:	0x00 00 00 04
0xFFFF 0004:	0x00 00 00 00
0xFFFF 0005:	0x00 00 00 05
0xFFFF 0007:	0x00 00 00 06
0xFFFF 0008:	0x00 00 00 07
0xFFFF 0009:	0x00 00 00 08

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

```
0x0000 FFFF: LOAD R1, R8;
0x0001 0000 ADD R28, R1, R1;
0x0001 0001: BZ R13;
0x0001 0002: ADD R11, R1, R11;
0x0001 0003: INC R8, R8;
0x0001 0004: BNZ R0;
0x0001 0005: STORE R11, R8;                :
```

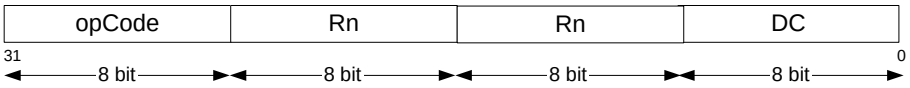
Forklar hva som skjer i koden. Hvilken verdi vil R11 ha etter at koden har kjørt?

e)
I en anna kjøring blir alle verdier (register og minne) satt tilbake til gitt utgangspunkt med unntak av R8 som nå blir gitt verdien: 0xFFFF 0004. Hvilken verdi vil R11 ha viss programmet nå blir kjørt på ny?

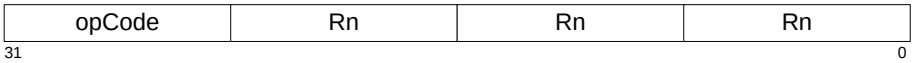
Load/store:

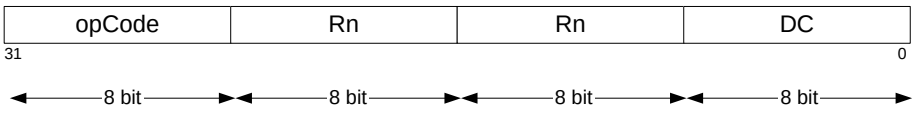
Load:

Store:



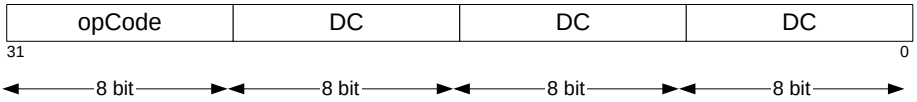
ALU:



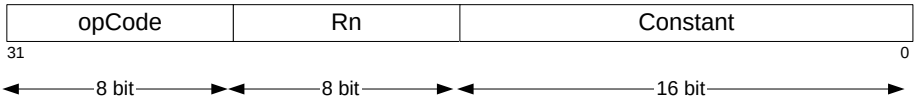


Spesial:

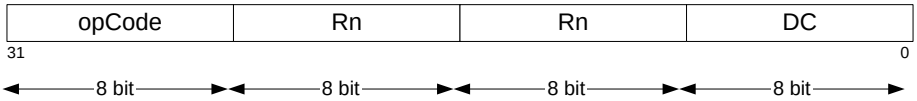
NOP:



MOVC:

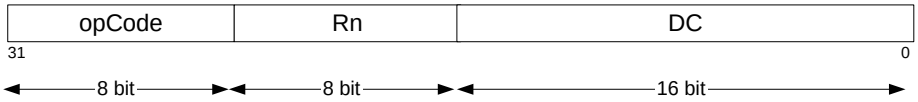


CP:

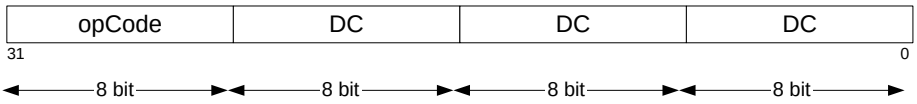


Flow control:

BZ/BNZ



RT



Rn: any user register, R0 - R31
DC: Don’t care: any data memory location

Figur 6 Instruction format Ram 007

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.

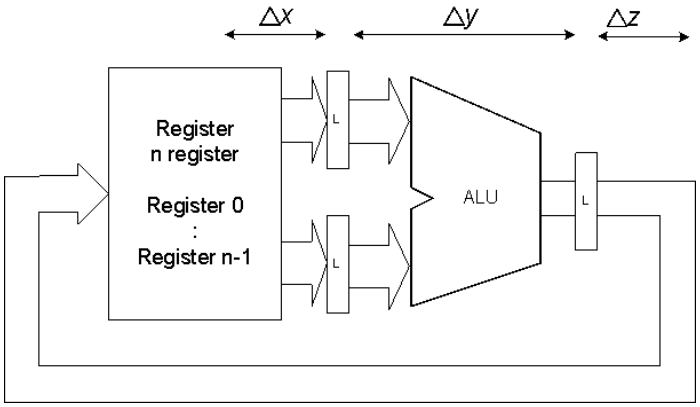
BZ, Ri Conditional branch on zero (Z-flag = 1) , $PC = Ri$.
BNZ, Ri Conditional branch on non zero (Z-flag = 0), $PC = Ri$.
RT Return, return from branch.

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

Figur 7 Instruction set RaM 007.

Oppgave 5 Ytelse (20 % (a: 5 %, b: 10 %, 2,5 % på c og 2,5 på d))

a)
Figur 8 viser en data path som inkluderer en registerbank, en ALU og tre latch-er. $\Delta x = 10ns$, $\Delta y = 25ns$ og $\Delta Z = 12ns$. Ut fra tilgjengelig informasjon, hvilket parameter begrenser klokkefrekvensen, og hva er maksimum klokkefrekvens? Forklar svaret kort.



Figur 8 Data path.

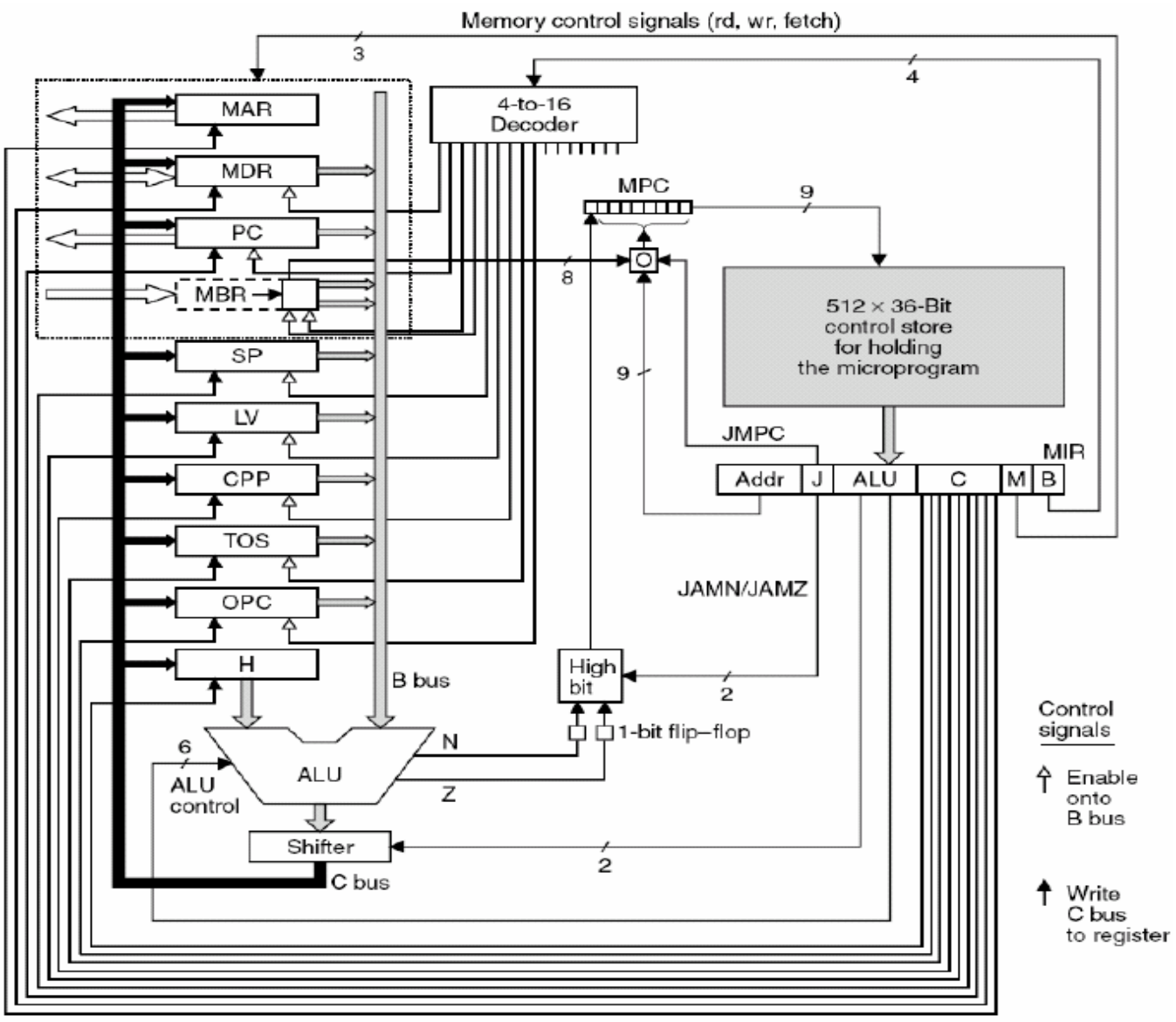
b)
1) Figur 9 viser den originale IJVM. Figur 13 viser en versjon av IJVM der det er innført en ekstra buss (A bus) og en Instruction Fetch unit for å øke ytelsen.

i) Hvordan påvirker innføringen av den ekstra bussen (A bus) instruksjonsutføringen? Forklar kort.
ii) Må mikroprogrammene for instruksjonene i control store endres for å utnytte den ekstra bussen? Forklar kort.

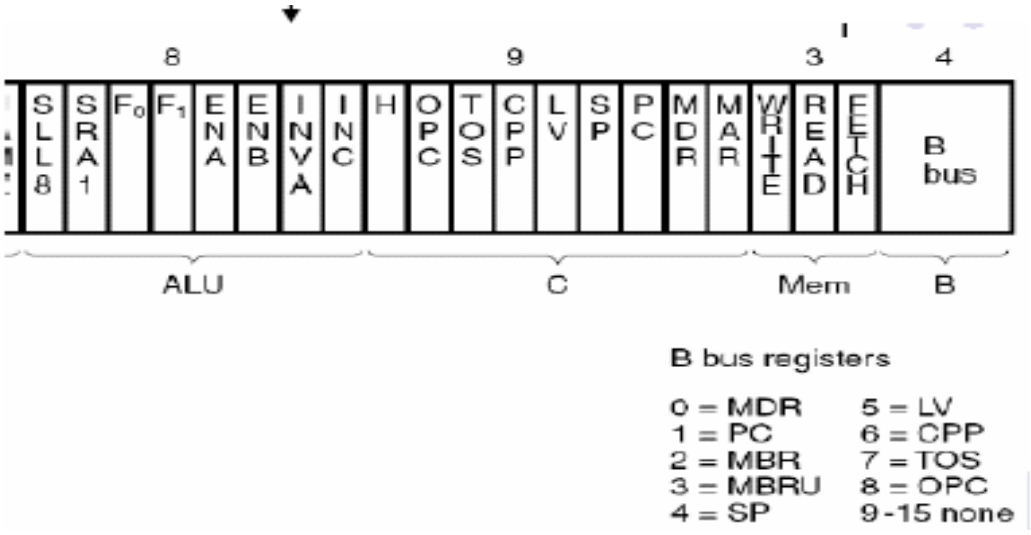
c)
Hvilken utvikling skisserer «Moore's law»?

d)
Hvilket parameter kunne man redusere for å få ned energiforbruket ved å gå fra prosessorer med en kjerne med dype samlebånd (pipelines) til CMP-baserte prosessorer?

Vedlegg IJVM



Figur 9 IJVM

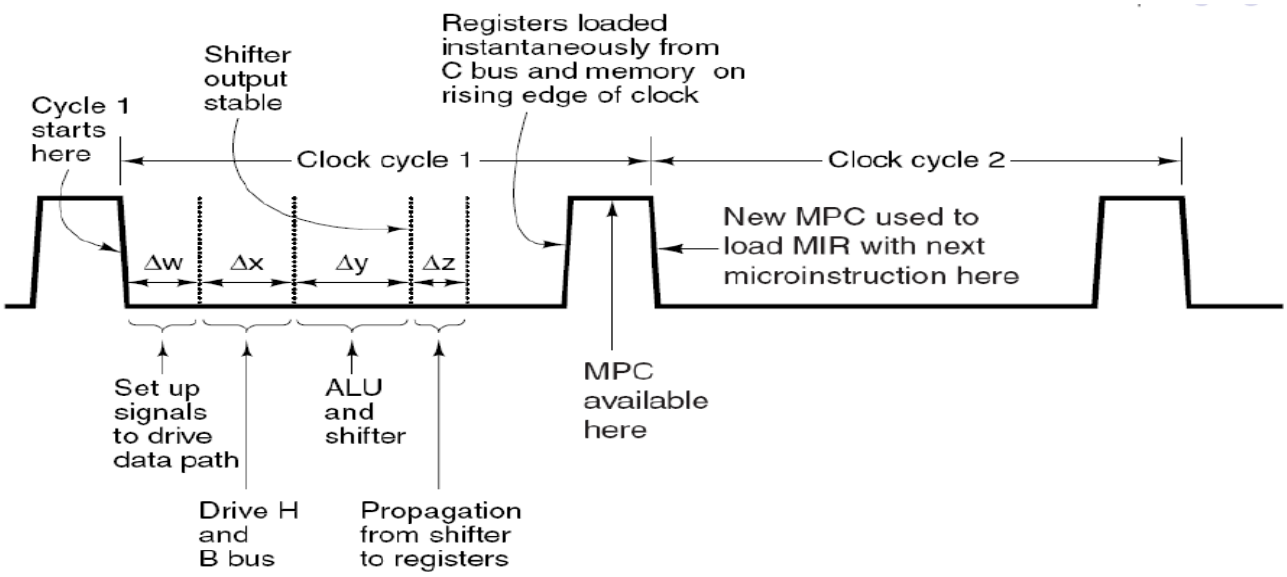


Figur 10 Microinstruction format.

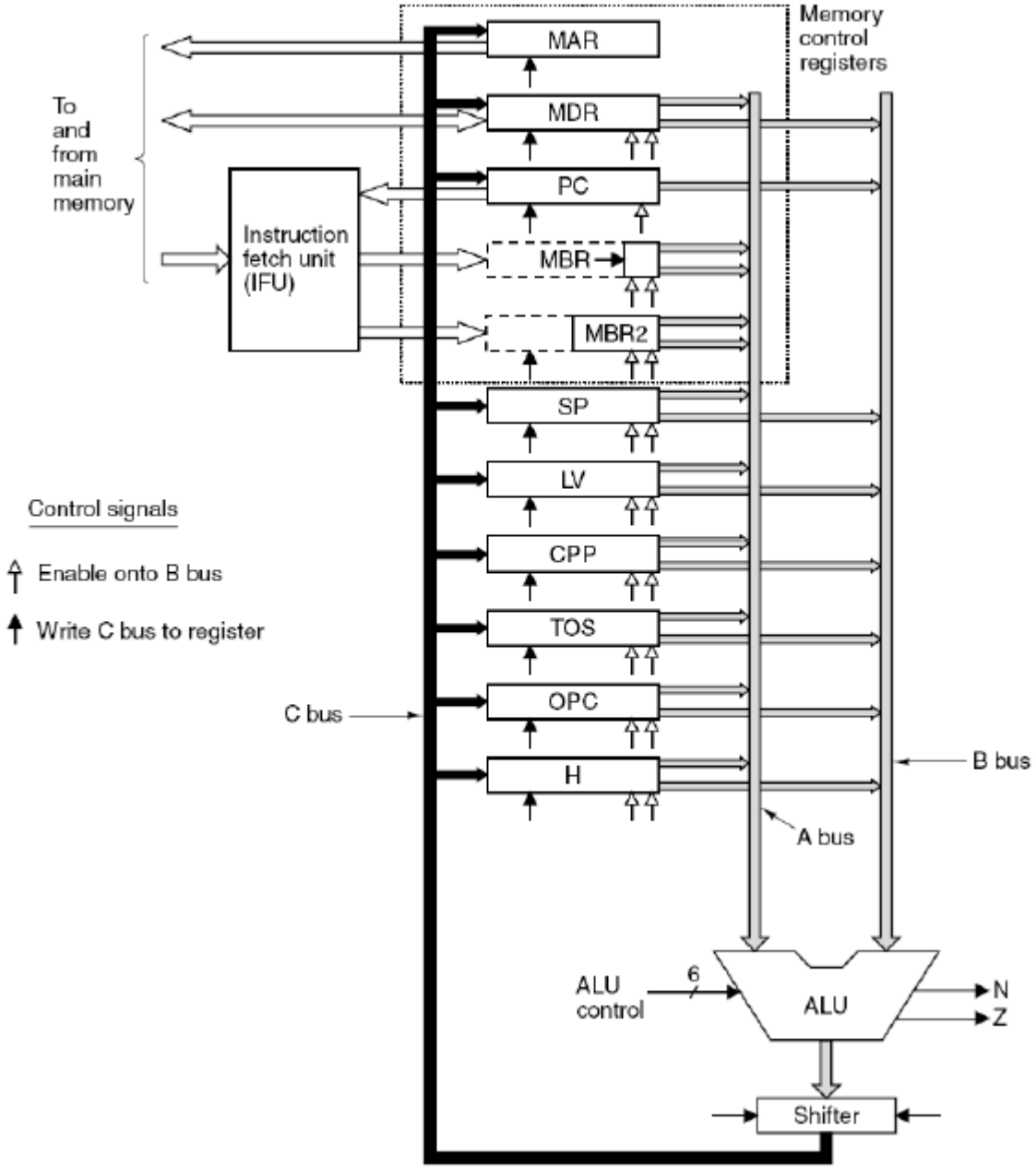
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	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 11 ALU functions.



Figur 12 Timing diagram.



Figur 13 MIC 2.