Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering
The Department of Computer and Information Science

TDT4160
DATAMASKINER GRUNNKURS
EKSAMEN

2. DESEMBER, 2011, 09:00–13:00

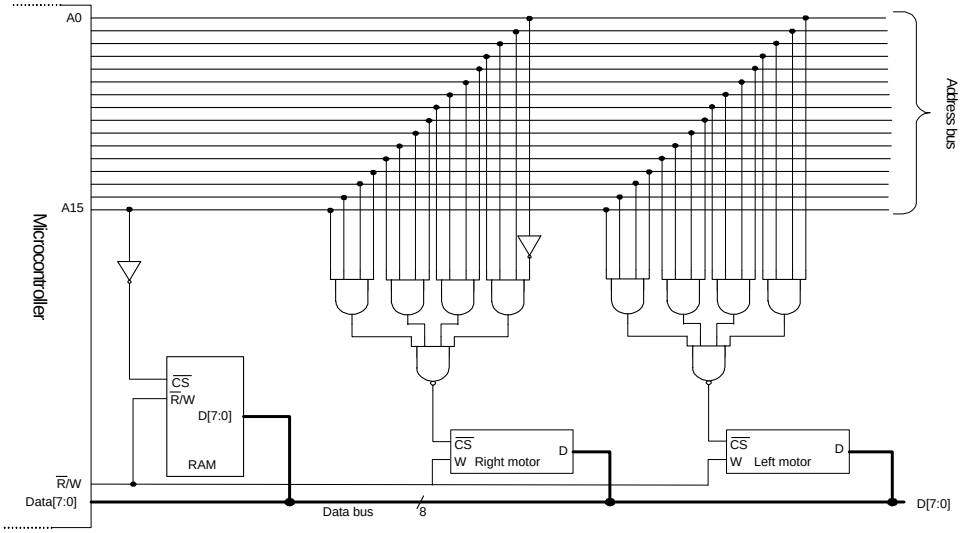
Kontakt under eksamen:
Gunnar Tufte 73590356/97402478

Tillate hjelpemidler:
D.
Ingen trykte eller håndskrevne hjelpemidler tillatt.
Bestemt, enkel kalkulator tillatt.

Målform:
Bokmål

OPPGAVE 1: DIGITAL LOGISK NIVÅ (20% (10% PÅ A, 5% PÅ B OG C))

a. I figur 1 er det eksterne bussgrensesnittet for en mikrokontroller vist. Mikrokontrolleren styrer en liten robot. RAM og to register for motorstyring deler en felles buss. RAM og de to motorstyringsregisterene har et aktivt lavt (logisk "0") CS (Chip Select) signal. Hva er adresseområdet til enhetene?



Figur 1: Address decoding.

b. Det totale adresserommet er på 64kByte. Størrelsen til RAM-en er 1kByte. Er det mulig å utvide med en ekstra RAM-modul med størrelse 32kByte, uten å endre på den eksisterende logikken?

c. Er systemet implementert med en multiplekset buss? Forklar.

OPPGAVE 2: MIKROARKITEKTUR OG MIKROINSTRUKSJONER (25% (5% PÅ A, B OG C; 10% PÅ D)))

Bruk vedlagte diagram i figur 6, figur 7, figur 8, figur 9, figur 10 og figur 11 for IJVM til å løse oppgavene.

a. Forklar funksjonelle forskjeller på "MPC" og "PC".

b. Lag mikroinstruksjon(er) for følgende IJVM-operasjon: last register "OPC" med summen av innholdet i register "TOS' og "LV" (*OPC = TOS + LV*).

   Se vekk fra Addr- og J-felta i mikroinstruksjonsformatet. Angi korrekte bit for ALU, C, Mem og B gitt i figur 7.

c. Hvilke endringer vil være nødvendige i styreenheten i IJVM-maskinvaren for å kunne styre den "data path" som er vist i figur 10?

d. I "control store" for IJVM vist i figur 6 ligger følgende mikroinstruksjoner for den tenkte instruksjonen IMassiveAdd:

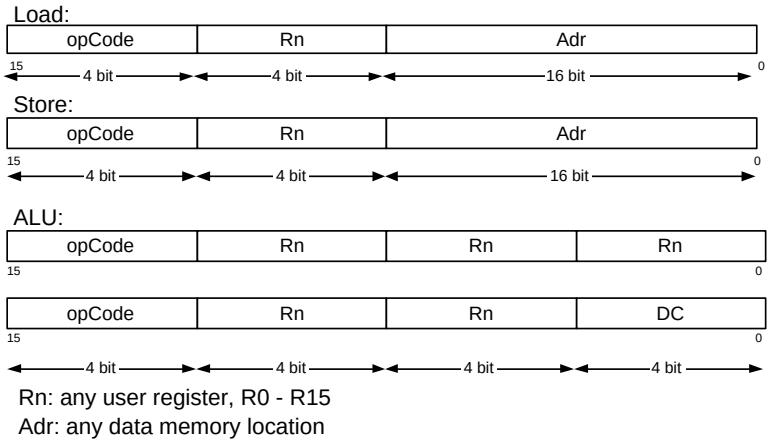
   MI 1: H = OPC
   MI 2: H = TOS + H
   MI 3: H = CPP + H
   MI 4: H = LV + H
   MI 5: OPC = SP + H

   Hver mikroinstruksjon i IMassiveAdd tar en klokkperiode å utføre, altså 5 klokkeperioder totalt. For å øke ytelsen blir mikroarkitekturen endret til den mikroarkitekturen som er vist i Figur 10.

   i) Hvor mange klokkeperioder kan utførelsen av IMassiveAdd nå reduseres til? Forklar.
   ii) Angi hvordan den opprinnelige IMassiveAdd kan modifiseres for å utnytte den nye mikroarkitekturen.

OPPGAVE 3: INSTRUKSJONSSETT ARKITEKTUR (ISA)(20% (7.5% PÅ A OG B; 5% PÅ C))

En veldig enkel prosessor har en "load", en "store" og 6 ALU-instruksjoner. Instruksjonsformatet for instruksjonene er vist i figur 2.



Figur 2: Instruction formats.

Alle instruksjonene i instruksjonssettet er:

LOAD: Load data from memory.

loadD Rn, Adr Load register Rn from memory location Adr.

STORE: Store data in memory.

storeD Rn, Adr Store register Rn in memory location Adr.

ALU: Data manipulation, register–register operations.

ADD Rn, Rn, Rn Bitwise AND, Rn = Rn · Rn.

NAND Rn, Rn, Rn Bitwise NAND, Rn = $\overline{Rn \cdot Rn}$.

OR Rn, Rn, Rn Bitwice OR, $Rn = Rn + Rn$.

INV Rn, Rn Bitwice invert, $Rn = \overline{Rn}$.

INC Rn, Rn Increment, $Rn = Rn + 1$

DEC Rn, Rn Decrement, $Rn = Rn - 1$

Rn any user register, *Adr* any data memory location.

a. Ut fra tilgjengelig informasjon, kan denne prosessoren defineres som generell? Forklar.

b. Prosessoren har ingen subtraksjonsinstruksjon, hvordan kan en subtraksjon utføres på denne prosessoren?

c. Hva er det totale minneområdet prosessoren kan direkte aksessere?

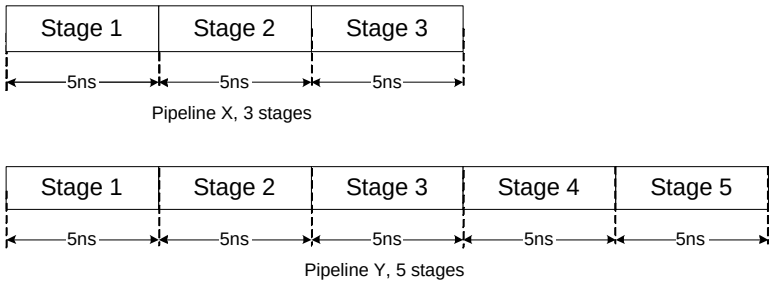
OPPGAVE 4: DATAMASKINER OG YTELSE (25% (5% PÅ A; 10% PÅ B OG C))

a. Figur 10 og figur 11 i vedleggene viser forskjellige versjoner av IJVM-mikroarkitekturer. Den opprinnelige mikroarkitekturen er vist i figur 6.

   Hvilke av de følgende tiltak for å øke ytelsen er gjort i de to mikroarkitekturene vist i figur 10 og figur 11?

   i) Superscalar arkitektur.
   ii) Pipelining.
   iii) Mer effektiv instruksjonsutførelse (mindre antall klokkeperioder).

b. En tenkt prosessor, HALL 9000, er designet med en tretrinns pipeline. Pipeline X i figur 3 viser denne tretrinns pipelinen. Forsinkelsen i hvert trinn er oppgitt. I et forsøk på å øke ytelsen til HALL 9000, gjøres det et redesign der det brukes en femtrinns pipeline. Pipeline Y i figur 3 viser den nye pipelinen med oppgitt trinnforsinkelse.



Figur 3: Pipeline designs for the HALL 9000 processor

   i) Hvordan påvirkes ytelsen til HALL 9000 av dette nye designet?
   ii) Hvordan påvirkes ILP (Instruction Level Parallelism) av redesignet?

c. Et lite testprogram brukes til å evaluere den nye versjonen av HALL 9000. Programmet er slik at ingen ”hazard”, ”pipeline flushes” eller dataavhengigheter oppstår. Programmet bruker 100 klokkeperioder på tretrinns pipelineutgaven av HALL 9000. Bruk tilgjengelig informasjon til å svare på følgende spørsmål:

   i) Hva er minimum antall klokkeperioder programmet kan utføres på hvis det kjøres på femtrinns utgaven av HALL 9000?

   ii)) Hva er minimal utførelsetid på HALL 9000 med tretrinns pipeline?

   iii) Hva er minimal utførelsetid på HALL 9000 med femtrinns pipeline?

OPPGAVE 5: DIVERSE BINÆR-QUIZ(10%)

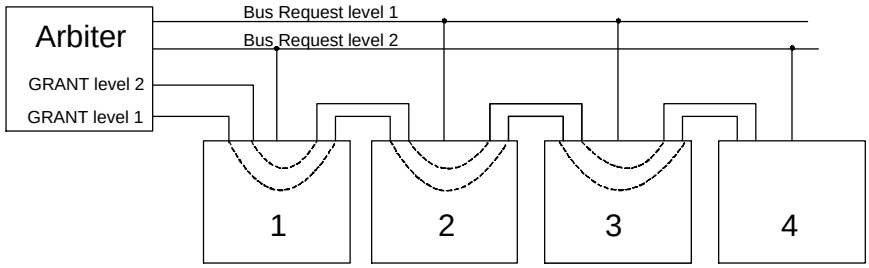
Velg riktig svar, sant eller usant. Korrekt svar gir 2%, feil -1%, blank eller flere svar på en oppgave gir 0%.

a. Ved "immediate addressing" inneholder instruksjonen også operanden (opcode og operand). Sant eller usant?

b. En Chip Multi Processor (CMP) er av type homogeneous eller heterogeneous. Sant eller usant?

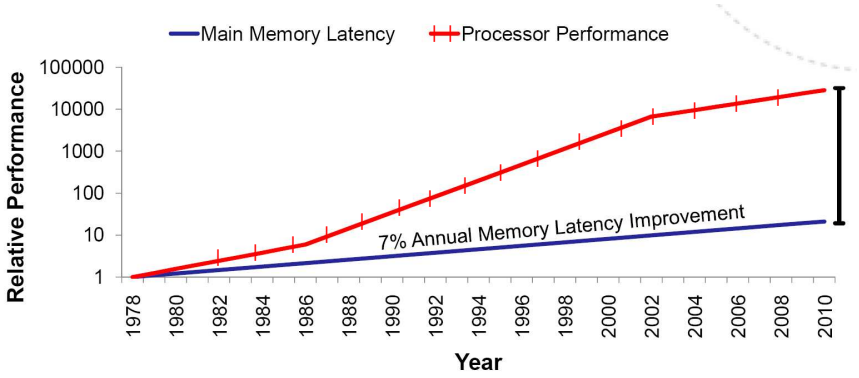
c. En statisk RAM-celle (SRAM) kan implementeres med færre transistorer enn en dynamisk RAM (DRAM)-celle. Sant eller usant?

d. Enhet 2 i figur 4 har høyest prioritet (level 1 er høyest). Sant eller usant?



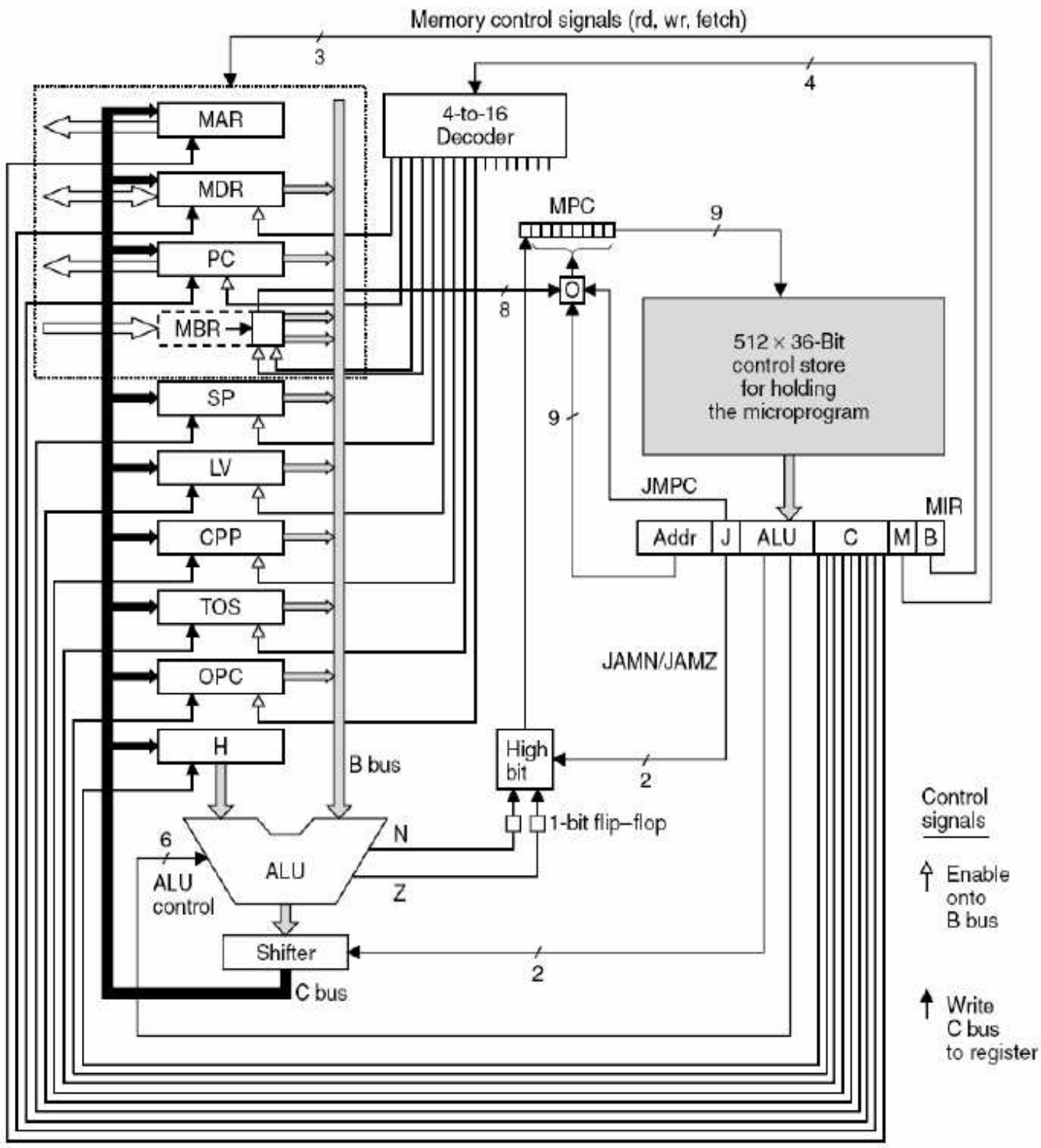
Figur 4: Centralized bus arbiter.

e. Cache hjelper til å skjule "the processor memory gap" —se figur 5. Sant eller usant?

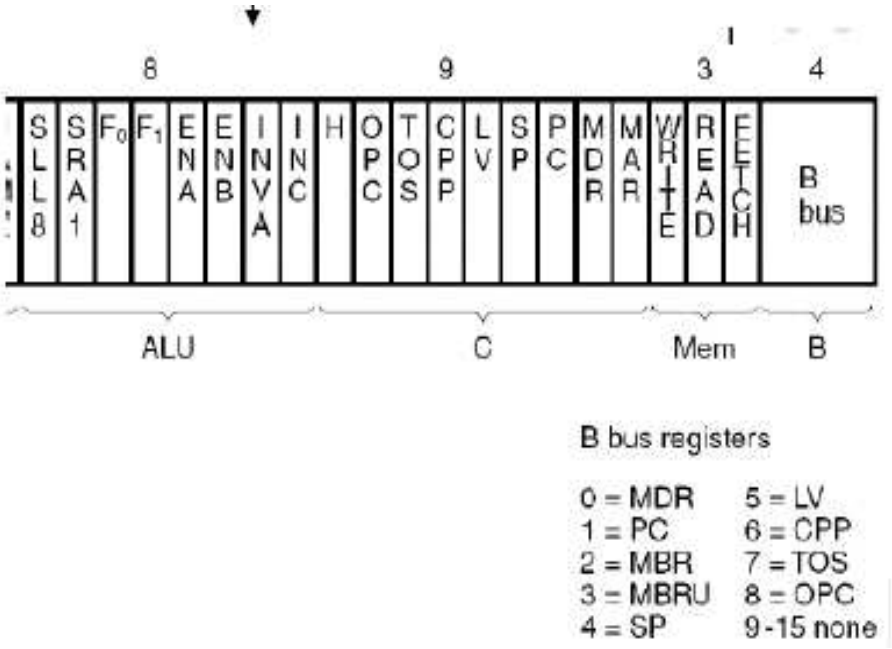


Figur 5: Processor memory gap.

IJVM vedlegg



Figur 6: Blokkdiagram (IJVM).



Figur 7: Mikroinstruksjonsformat (IJVM).

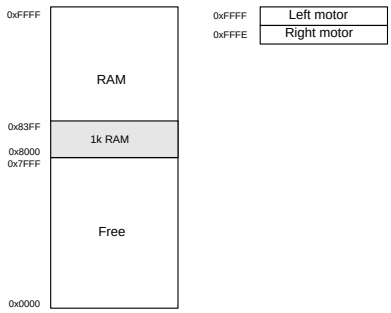
Answer Key for The Exam

OPPGAVE 1: DIGITAL LOGISK NIVÅ (20% (10% PÅ A, 5% PÅ B OG C))

a. I figur 1 er det eksterne bussgrensesnittet for en mikrokontroller vist. Mikrokontrolleren styrer en liten robot. RAM og to register for motorstyring deler en felles buss. RAM og de to motorstyringsregisterene har et aktivt lavt (logisk "0") CS (Chip Select) signal. Hva er adresseområdet til enhetene?

Answer: For full, angi adresse område.

Adresse område for einingane er:
Ledig: hex(0000) - hex(7FFF)
RAM: hex(8000) - hex (FFFF)
Right motor: hex(FFFE)
Left Motor: hex(FFFF)



Adresse område (memory map). Kunn fyrste RAM-avbilding vist.

RAM-adresseområde overlappar med adresseområde til motorregistera. Det gir ikkje elektriske problem sidan motorregistera kunn kan skrivast til.

Ekstra forklaring:
Ram brikken er berre på 1k, sidan det nyttast delvis adressedekoding vil ei kvar adresse over 0x8000 være ein RAM-aksess. Ved skriving til motorregister *Right motor* vil det også bli skrevet til minnelokasjon 0x3FE i RAM-brikken. Ved skriving til motorregister *left motor* vil det også bli skrevet til minnelokasjon 0x3FF i RAM-brikken,

b. Det totale adresserommet er på 64kByte. Størrelsen til RAM-en er 1kByte. Er det mulig å utvide med en ekstra RAM-modul med størrelse 32kByte, uten å endre på den eksisterende logikken?

Answer: I adresserommet er det nokk ledig plass (0x0000 - 7FFF) til ein RAM-brikke på 32k.

c. Er systemet implementert med en multiplekset buss? Forklar.

Answer: Nei, det er separat databuss og adressebuss.

OPPGAVE 2: MIKROARKITEKTUR OG MIKROINSTRUKSJONER (25% (5% PÅ A, B OG C; 10% PÅ D)))

Bruk vedlagte diagram i figur 6, figur 7, figur 8, figur 9, figur 10 og figur 11 for IJVM til å løse oppgavene.

a. Forklar funksjonelle forskjeller på "MPC" og "PC".

Answer: **MPC:** MicrProgramCounter peikar på microinstruksjon i "control store". Ved instruksjonsutføring peikar MPC på fyrste micro instruksjon i aktiv instruksjon. Updaterast med adressa til neste micro instruksjon i instruksjonen, til alle micro instruksjonar i instruksjonen er utført (etter siste micro instruksjon set til micro instruksjon for PC oppdatering. Sjå boka for utfølgjande info. Treng berre kva det er og at MPC held orden på kva som er neste micro instruksjon for max utteljing.

PC: Program Counter peikar på INSTRUKSJON i programminne, opcoden til instruksjonen blir lasta i MBR og overført til MPC.

b. Lag mikroinstruksjon(er) for følgende IJVM-operasjon: last register "OPC" med summen av innholdet i register "TOS' og "LV" (*OPC = TOS + LV*).

Se vekk fra Addr- og J-felta i mikroinstruksjonsformatet. Angi korrekte bit for ALU, C, Mem og B gitt i figur 7.

Answer: Kan gjerast på to (minst) forskjelige måtar:
H = LV
OPC = TOS + H
ALU: 010100 (B) C: 100000000 (H) Mem: 000 (ingen mem. oppr.) B: 0101 (LV)
ALU: 111100 (A + B) C: 010000000 (OPC) Mem: 000 (ingen mem. oppr.) B: 0111 (TOS)
eller:
H = TOS
OPC = LV + H
ALU: 010100 (B) C: 100000000 (H) Mem: 000 (ingen mem. oppr.) B: 0111 (TOS)
ALU: 111100 (A + B) C: 010000000 (OPC) Mem: 000 (ingen mem. oppr.) B: 0101 (LV)

c. Hvilke endringer vil være nødvendige i styreenheten i IJVM-maskinvaren for å kunne styre den "data path" som er vist i figur 10?

Answer: Control store må utvidast til å includere eit felt for å styre *A-bus*, MIR må også utvidast for å kunne halde dei ekstra styre signala. For å styre bussen effektivt kan ein utvide med ein ekstra 4 til 16 dekoder (som for B-bus) då vil utvidinga krevve 4 ekstra bit i mikroinstruksjonar. i.e. control store må vere 512 x 40 bit og MIR må utvidast tilsvarande med 4 bit for A-buss.

d. I "control store" for IJVM vist i figur 6 ligger følgende mikroinstruksjoner for den tenkte instruksjonen IMassiveAdd:

MI 1: H = OPC
MI 2: H = TOS + H
MI 3: H = CPP + H

MI 4: H = LV + H

MI 5: OPC = SP + H

Hver mikroinstruksjon i IMassiveAdd tar en klokkperiode å utføre, altså 5 klokkeperioder totalt. For å øke ytelsen blir mikroarkitekturen endret til den mikroarkitekturen som er vist i Figur 10.

i) Hvor mange klokkeperioder kan utførelsen av IMassiveAdd nå reduseres til? Forklar.

ii) Angi hvordan den opprinnelige IMassiveAdd kan modifiseres for å utnytte den nye mikroarkitekturen.

Answer: IMassiveAdd gjer følgjande: OPC = OPC + TOS + CPP + LV + SP. Ved å innføre ekstar buss kan mikroinstruksjonane til IMassiveAdd gjerast meir effektive, i.e. færre microinstruksjonar, som gir færre klokke periodar for å utføre instruksjonen.
Forslag, mange måtar å løyse på, men noko slikt som dette gir full utteljing:

Antal klokke periodar kan reduserast med 1, til 4 totalt. Følgjand micrinstruksjonar angir korleis IMassiveAdd for arkitekturen i figur 10 kan implementerest:

MI 1: OPC = OPC + SP

MI 2: OPC = TOS + OPC

MI 3: OPC = CPP + OPC

MI 4: OPC = LV + OPC

Det eksisterar kanskje lurare løysingar, då gir dei også full utteljing. Men i løysinga vist er innhalde i TOS, CPP, LV og SP uendra etter at instruksjonen er ferdig (ikkje krav for å få full utteljing).

OPPGAVE 3: INSTRUKSJONSSETT ARKITEKTUR (ISA)(20% (7.5% PÅ A OG B; 5% PÅ C))

En veldig enkel prosessor har en "load", en "store" og 6 ALU-instruksjoner. Instruksjonsformatet for instruksjonene er vist i figur 2.
Alle instruksjonene i instruksjonssettet er:

LOAD: Load data from memory.

loadD Rn, Adr Load register Rn from memory location Adr.

STORE: Store data in memory.

storeD Rn, Adr Store register Rn in memory location Adr.

ALU: Data manipulation, register–register operations.

ADD Rn, Rn, Rn Bitwise AND, Rn = Rn · Rn.

NAND Rn, Rn, Rn Bitwise NAND, Rn = $\overline{Rn \cdot Rn}$.

OR Rn, Rn, Rn Bitwice OR, $Rn = Rn + Rn$.

INV Rn, Rn Bitwice invert, $Rn = \overline{Rn}$.

INC Rn, Rn Increment, $Rn = Rn + 1$

DEC Rn, Rn Decrement, $Rn = Rn - 1$

Rn any user register, *Adr* any data memory location.

a. Ut fra tilgjengelig informasjon, kan denne prosessoren defineres som generell? Forklar.

Answer: Nei, manglar instruksjon for branch (flow control). Dette er ein av dei fundamentale krava for at ei maskin kan være generell. Ei generell maskin må kunne utføre I/O operasjonar (load og store fiksar det), Datamanipulasjon (ALU-operasjonar dekkar det) og Branch (flyt controll (er ingen slike instruksjonar)).

Ekstra:
Dette er ei særs lite effektiv maskin, må f.eks. bruke logikske operasjonar for å flytte data frå eit register til eit anna.

Prosessoren har ingen subtraksjonsinstruksjon, hvordan kan en subtraksjon utføres på denne prosessoren?

Answer: I oppgåva er det ein trykk feil, fyrste ALU instruksjon heiter ADD, men gjer bitwise AND. Det er ikkje veldigt viktigt, men opnar for 3 forskjellige svar alternativ etter som korleis ein vel å tolke oppgåva, alle er rekna som heilt rett:

Alternativ 1: Viss det er valgt at ALU kan utføre ADD så er svaret at det er mulig sidan det er instruksjonar som kan utføre logikk for toerskomplement og så ein addisjon, noko slikt som denne koden:

INV Rx, Rx
INC Rx, Rx
ADD Rz, Ry, Rx
(Rz = Ry - Rx)

Kan sjølvsagt ta med minne instruksjonar også:
loadD Rx, Adr loadD Ry, Adr INV Rx, Rx
INC Rx, Rx
ADD Rz, Ry, Rx
storeD Rz, Adr

Alternativ 2: Viss det er valgt å bruke fyrste instruksjon som **Bitwise AND**:
Prosessoren har logiske operasjonar som kan utføre ein addisjon, prosessoren kan då også utføre subtraksjon ved hjelp av 2erskompliment.

Alternativ 3: Viss det er valgt å bruke fyrste instruksjon som **Bitwise AND**:
Opnar også for at prosessoren har logiske operasjonar, men ADD er ikkje mulig, så då er det ikkje mulig å utføre subtraksjon ved hjelp av 2erskompliment.

Oppgåva rettast med all mulig godvilje, men svaret må være fornuftigt grunna.

Hva er det totale minneområdet prosessoren kan direkte aksessere?

Answer: 2^{16} minne lokasjonar.

OPPGAVE 4: DATAMASKINER OG YTELSE (25% (5% PÅ A; 10% PÅ B OG C))

a. Figur 10 og figur 11 i vedleggene viser forskjellige versjoner av IJVM-mikroarkitekturer. Den opprinnelige mikroarkitekturen er vist i figur 6.

   Hvilke av de følgende tiltak for å øke ytelsen er gjort i de to mikroarkitekturene vist i figur 10 og figur 11?

   i) Superscalar arkitektur.

   ii) Pipelining.

   iii) Mer effektiv instruksjonsutførelse (mindre antall klokkeperioder).

   Answer: Ytelse forbedrings er som følgjer:

   i ingen av microarkitekturane er Superscalare.

   ii I figur 11 er det innført ei pipeline.

   iii I figur 10 er det mulig å ha meir effektiv microinstruksjonar (A-buss), denne ytelseforbedringen er også videreført i figur 11.

b. En tenkt prosessor, HALL 9000, er designet med en tretrinns pipeline. Pipeline X i figur 3 viser denne tretrinns pipelinen. Forsinkelsen i hvert trinn er oppgitt. I et forsøk på å øke ytelsen til HALL 9000, gjøres det et redesign der det brukes en femtrinns pipeline. Pipeline Y i figur 3 viser den nye pipelinen med oppgitt trinnforsinkelse.

   i) Hvordan påvirkes ytelsen til HALL 9000 av dette nye designet?

   ii) Hvordan påvirkes ILP (Instruction Level Parallelism) av redesignet?

   Answer: Dette var eit dårleg ytelseforbetringstiltak, brukar meir resursar og ytelsen går (litt) ned.

   i) Ytelsen vil gå litt ned sidan det tar lengre tid å tømme/fylle ei 5-stegs pipeline. Sidan max trinnforsinkelsen ikkje er minka vil klokkefrekvensen være den same.

   ii) ILP vil auke sidan prosessoren no prosesserar 5 instruksjonar i paralell.

c. Et lite testprogram brukes til å evaluere den nye versjonen av HALL 9000. Programmet er slik at ingen "hazard", "pipeline flushes" eller dataavhengigheter oppstår. Programmet bruker 100 klokkeperioder på tretrinns pipelineutgaven av HALL 9000. Bruk tilgjengelig informasjon til å svare på følgende spørsmål:

i) Hva er minimum antall klokkeperioder programmet kan utføres på hvis det kjøres på femtrinns utgaven av HALL 9000?

ii)) Hva er minimal utførelsetid på HALL 9000 med tretrinns pipeline?

iii) Hva er minimal utførelsetid på HALL 9000 med femtrinns pipeline?

Answer: Brukar tidene oppgitt og antal klokke periodar oppgitt til å finne svar.

i) 102, treng to meir for å fylle pipelina før fyrste instruksjon er ferdigt.

ii)) $100 * 5ns = 500ns$

iii) $102 * 5ns = 510ns$

OPPGAVE 5: DIVERSE BINÆR-QUIZ(10%)

Velg riktig svar, sant eller usant. Korrekt svar gir 2%, feil -1%, blank eller flere svar på en oppgave gir 0%.

(i) Ved "immediate addressing" inneholder instruksjonen også operanden (opcode og operand). Sant eller usant?

Answer: Sant

(ii) En Chip Multi Processor (CMP) er av type homogeneous eller heterogeneous. Sant eller usant?

Answer: Sant

(iii) En statisk RAM-celle (SRAM) kan implementeres med færre transistorer enn en dynamisk RAM (DRAM)-celle. Sant eller usant?

Answer: Usant

(iv) Enhet 2 i figur 4 har høyest prioritet (level 1 er høyest). Sant eller usant?

Answer: Sant (2, 3, 1, 4)

(v) Cache hjelper til å skjule "the processor memory gap" —se figur 5. Sant eller usant?

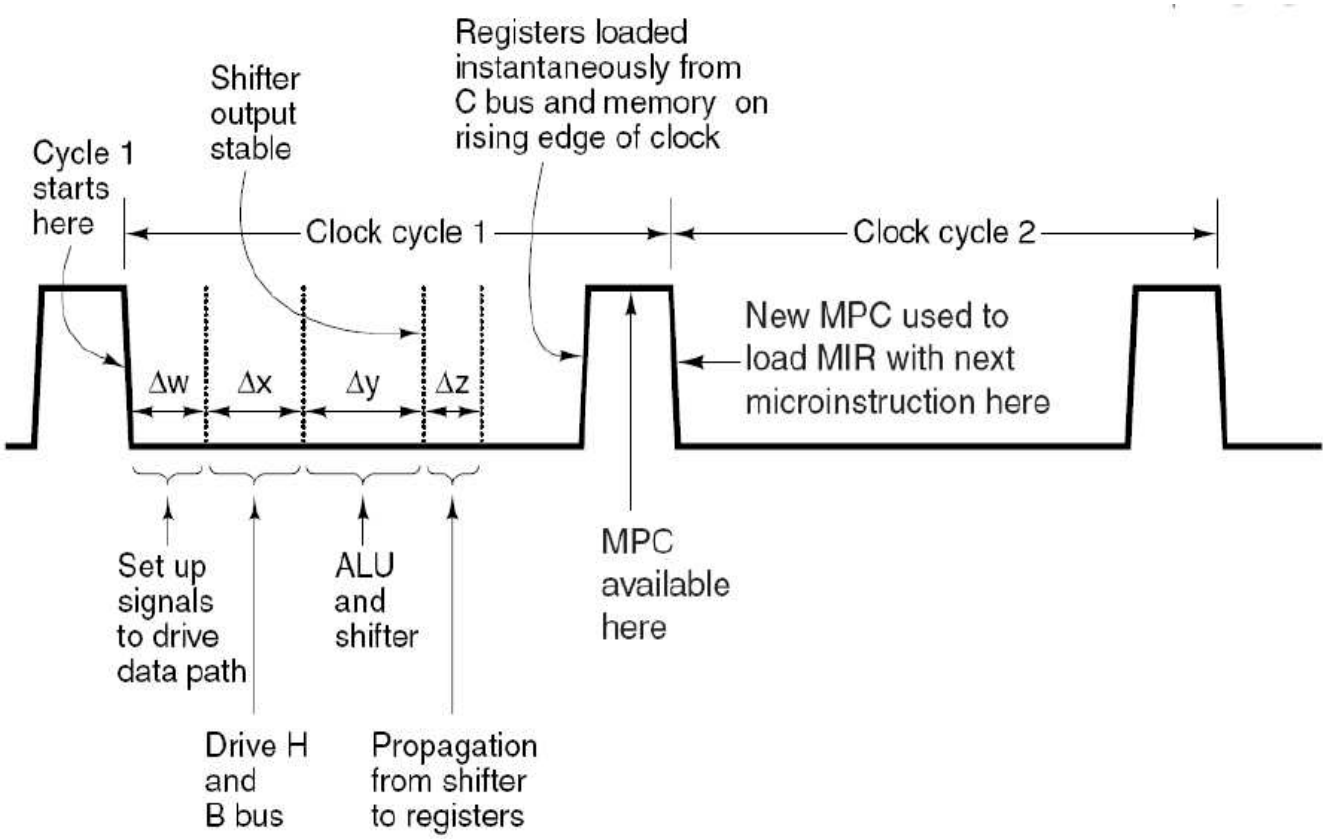
Answer: Sant

IJVM vedlegg

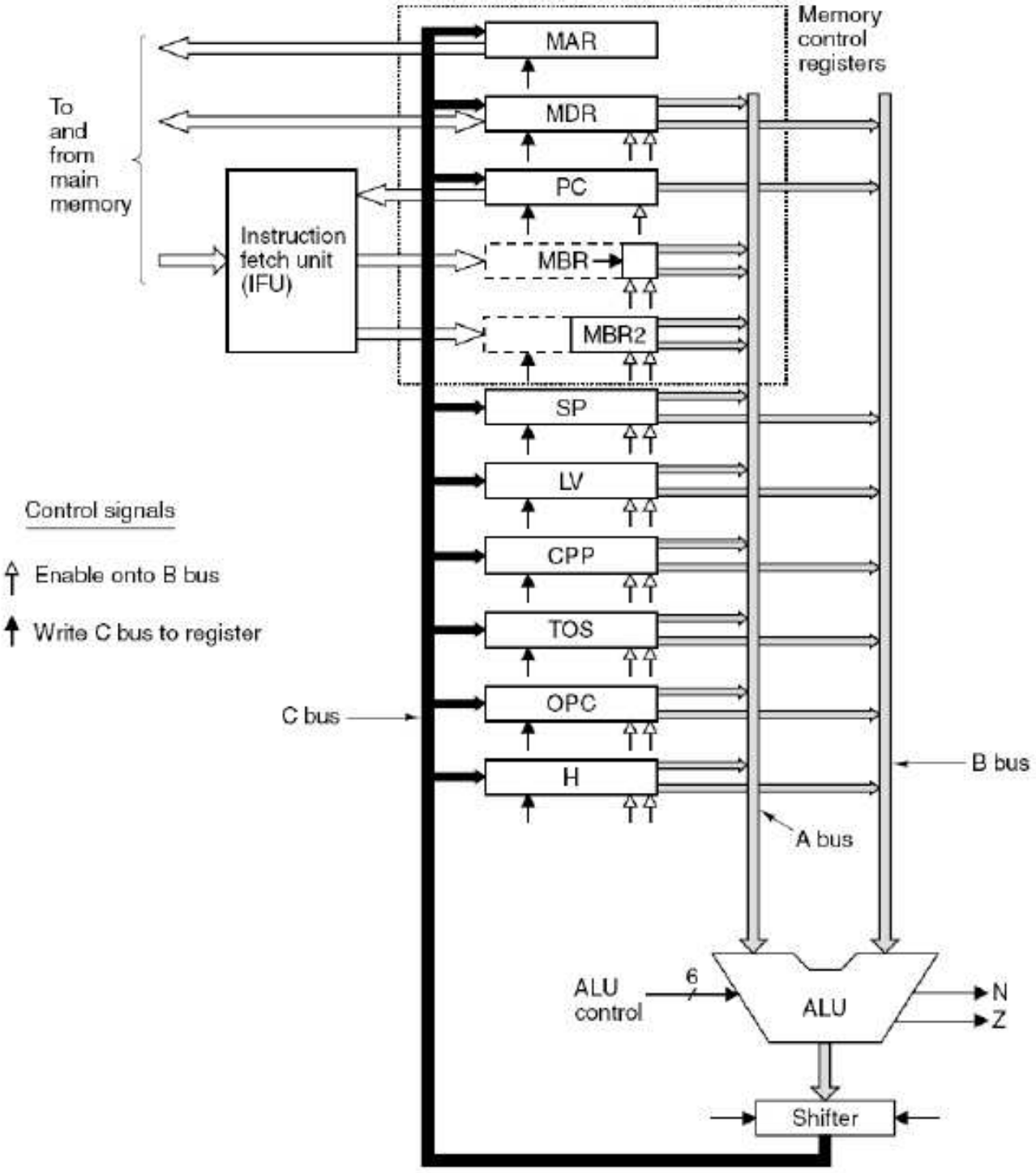
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

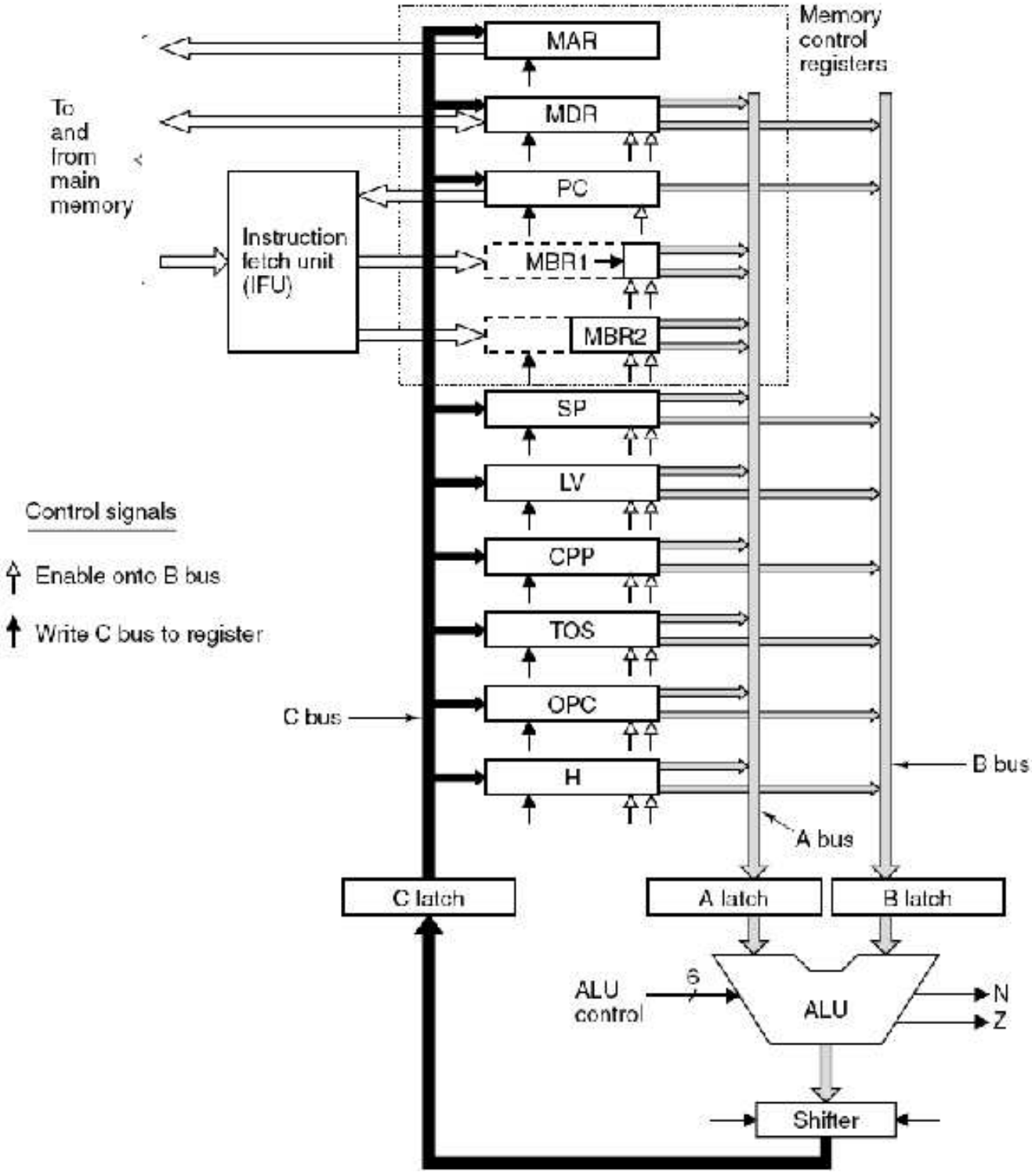
Figur 8: Funksjonstabell for ALU (IJVM).



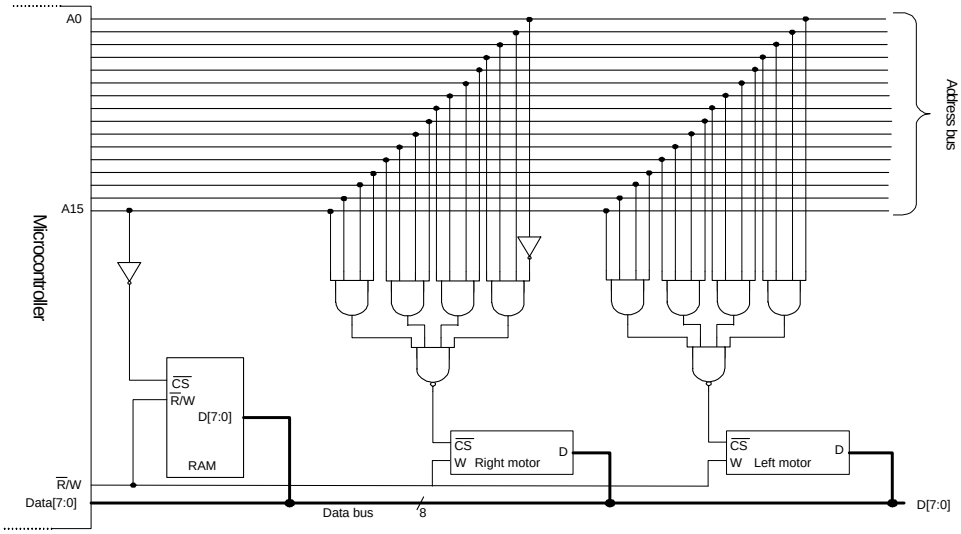
Figur 9: Timingdiagram (IJVM).



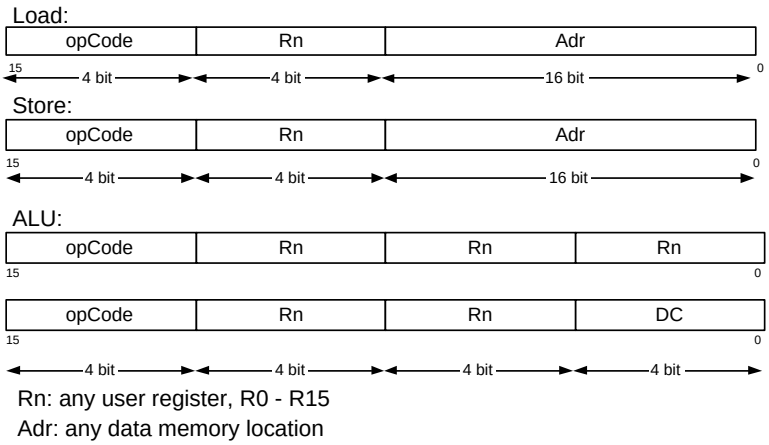
Figur 10: Alternativ mikroarkitektur I.



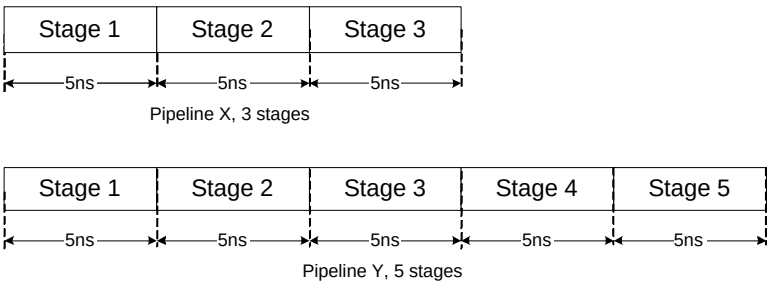
Figur 11: Alternativ mikroarkitektur II.



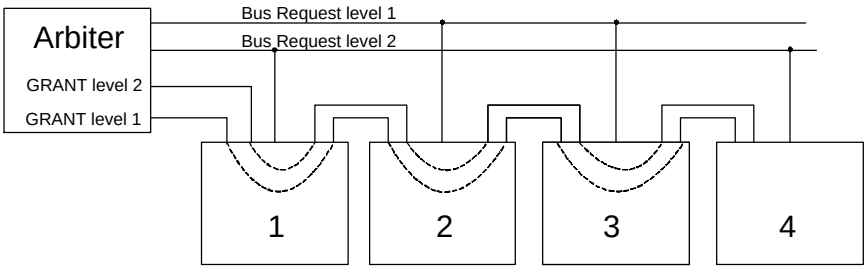
Figur 12: Address decoding.



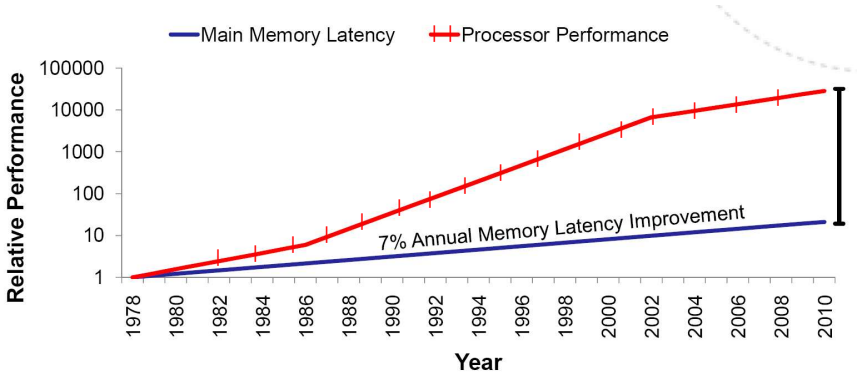
Figur 13: Instruction formats.



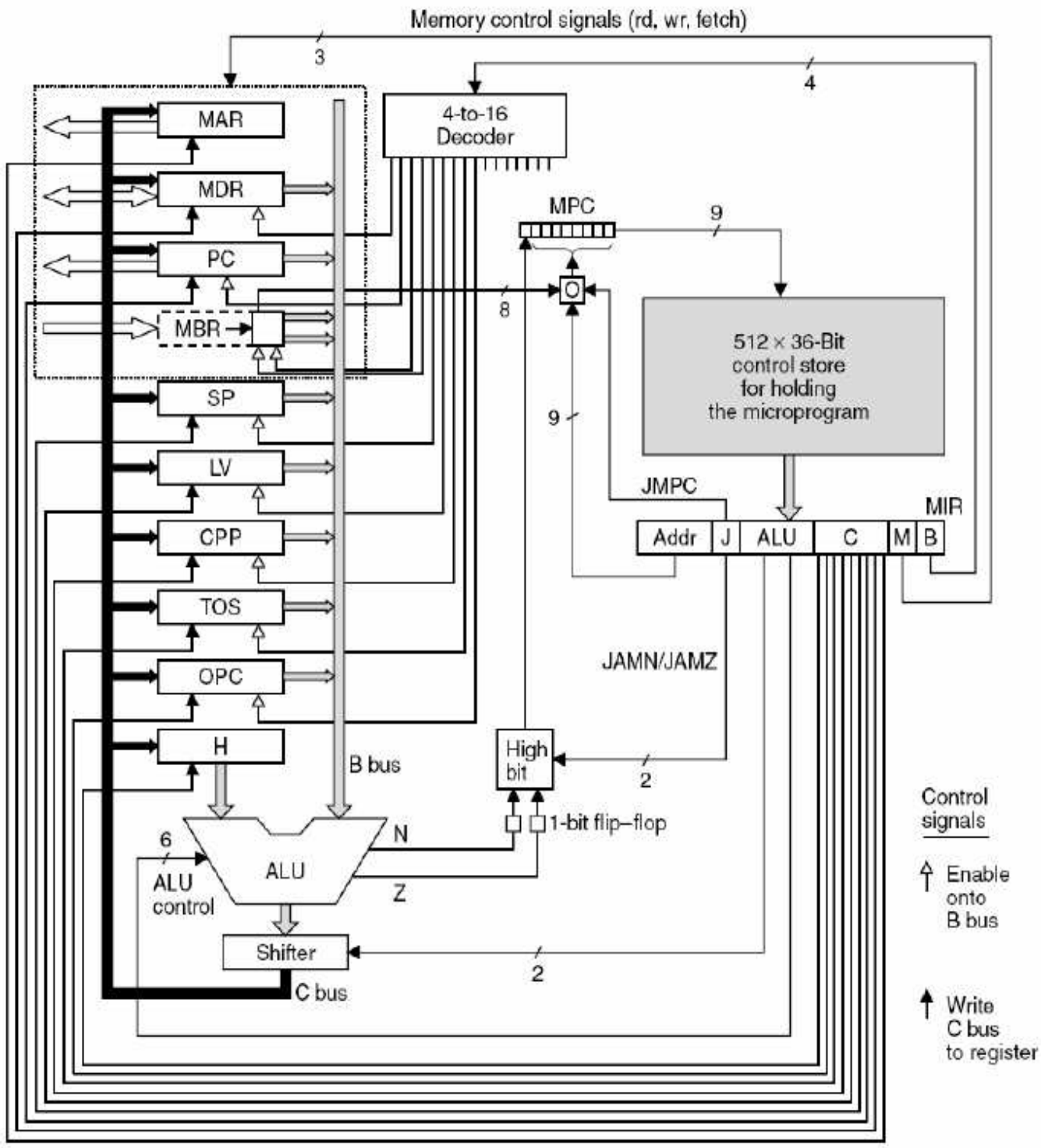
Figur 14: Pipeline designs for the HALL 9000 processor



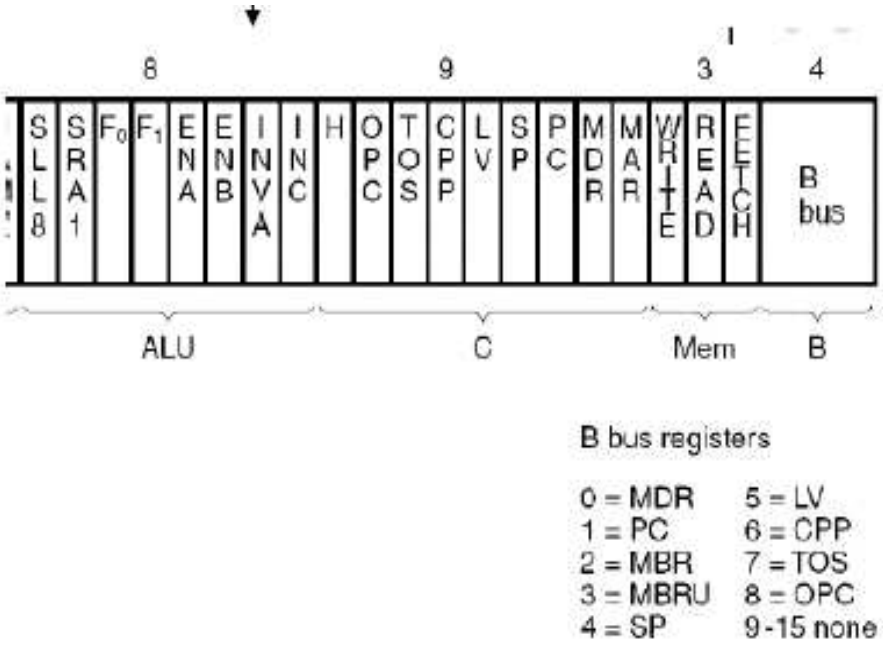
Figur 15: Centralized bus arbiter.



Figur 16: Processor memory gap.



Figur 17: Blokkdiagram (IJVM).

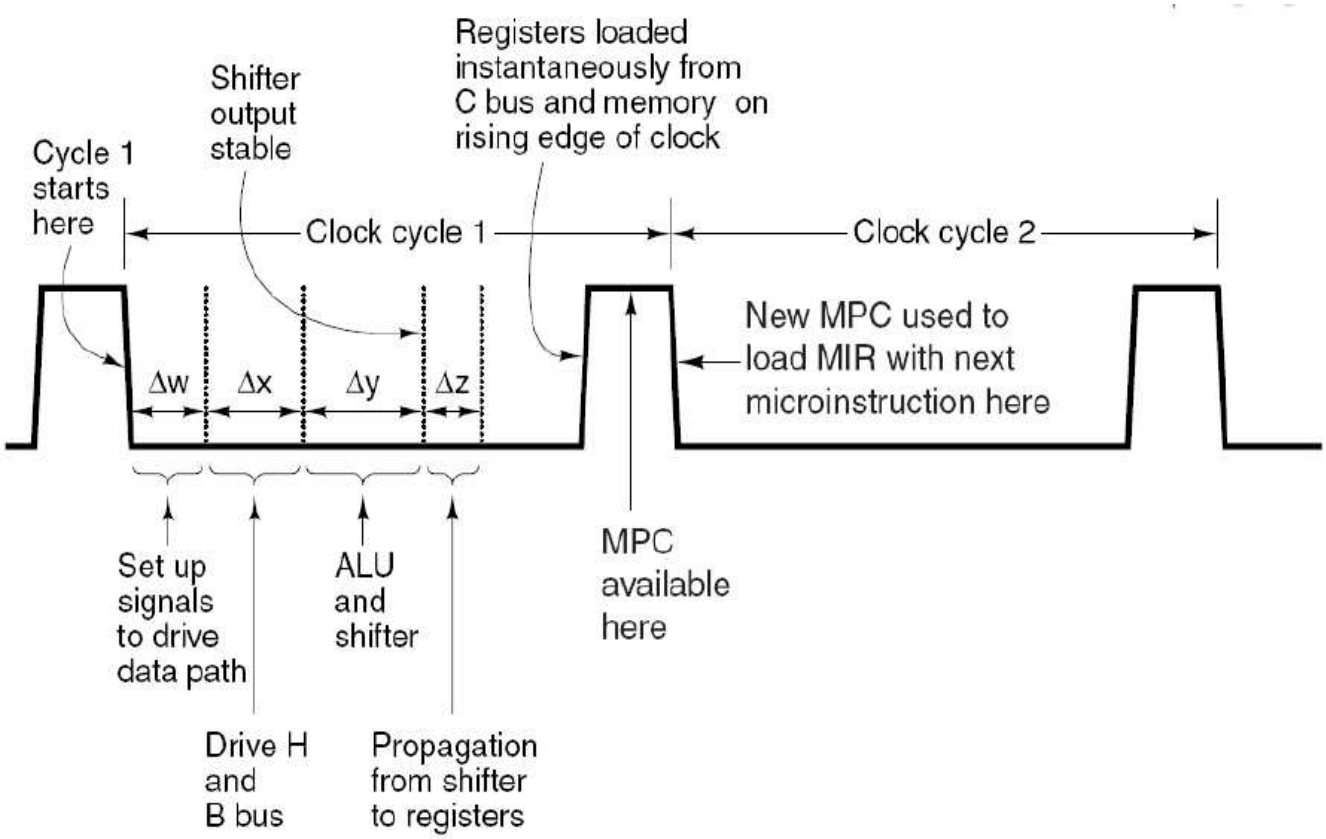


Figur 18: Mikroinstruksjonsformat (IJVM).

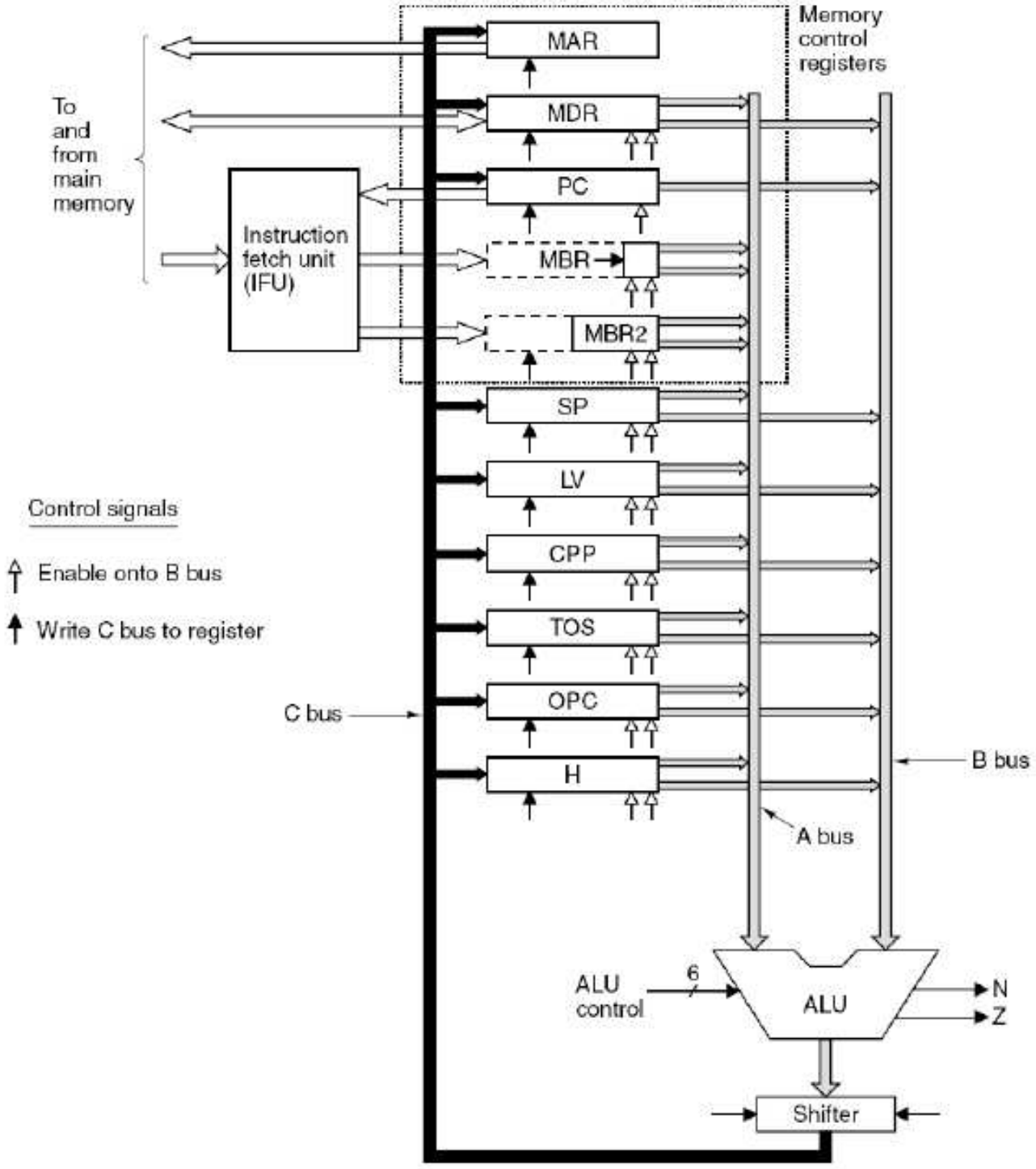
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

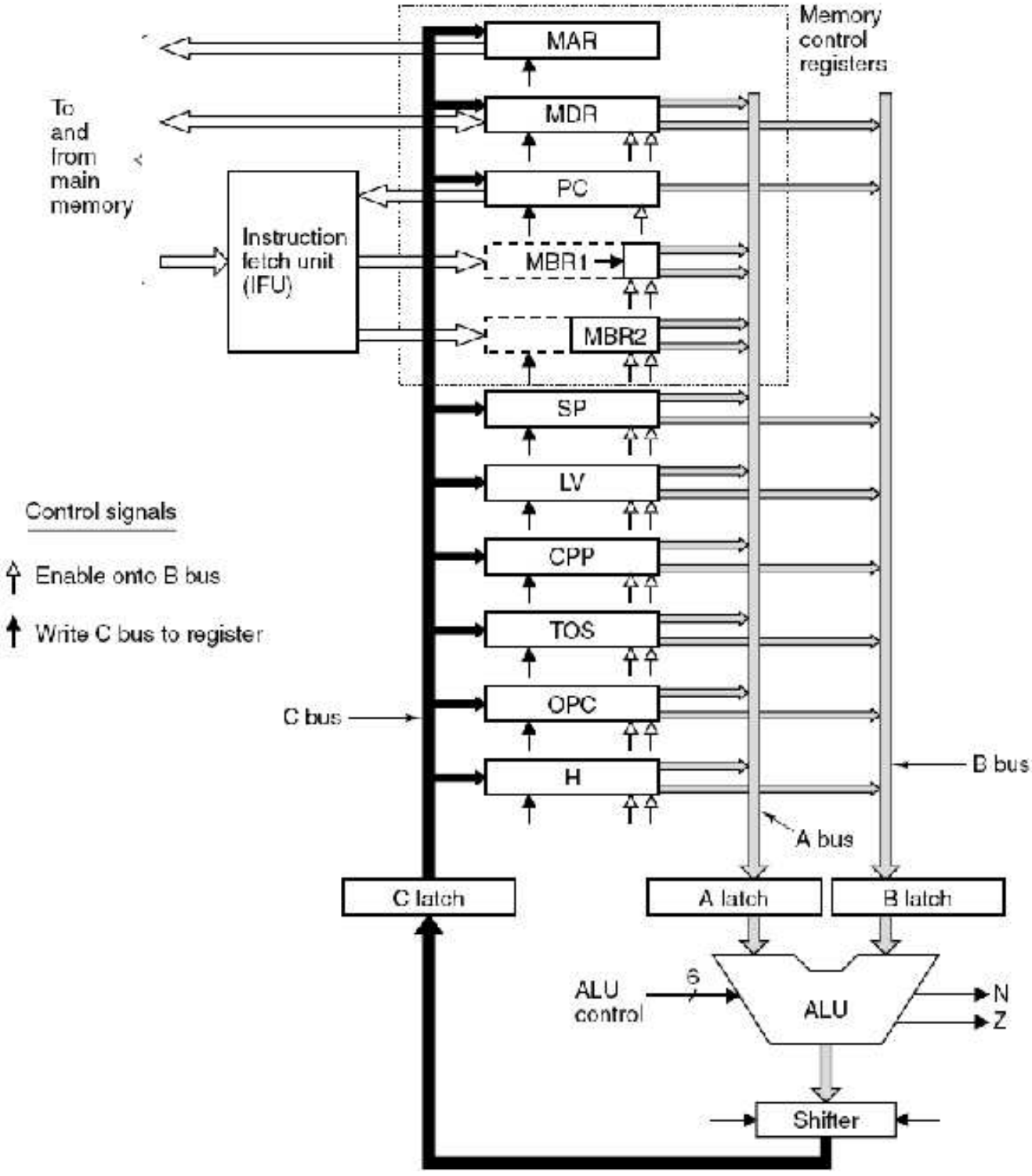
Figur 19: Funksjonstabell for ALU (IJVM).



Figur 20: Timingdiagram (IJVM).



Figur 21: Alternativ mikroarkitektur I.



Figur 22: Alternativ mikroarkitektur II.