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

TDT4160
DATAMASKINER GRUNNKURS
EKSAMEN

16. DESEMBER, 2013, 09:00–13:00

Kontakt under eksamen:
Gunnar Tufte 73590356/97402478

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

Målform:
Bokmål

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

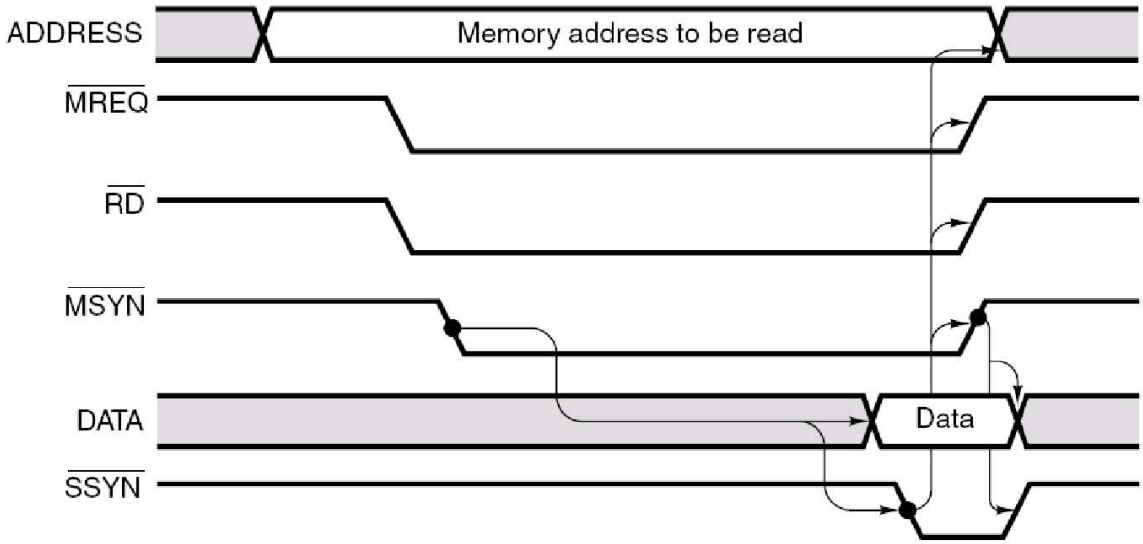
a. Unrer konstruksjon av et datamaskinsystem er det krav til minnesystemet om en gjennomsnittlig minneaksesstid (memory access time) på minimum 4.5ns. To forskjellige løsninger er tilgjengelig.

   Minnesystem-A har et nivå med hurtigbuffer (cache) med en aksesstid på 2ns og med et trefforholdstall (hit ratio) på 90 %. Hovedminnet har en aksesstid på 20ns.

   Minnesystem-B har et nivå med hurtigbuffer (cache) med en aksesstid på 1ns og med et trefforholdstall (hit ratio) på 60 %. Hovedminnet har en aksesstid på 15ns.

   Tilfredstiller noen av de to minnesystemene kravet om 4.5ns gjennomsnittlig minneaksesstid? I så fall hvilket?

b. Ut fra informasjonen i figur 1, er dette en asynkron eller synkron bussoverføring? Begrunn svaret kort.



Figur 1: Address decoding.

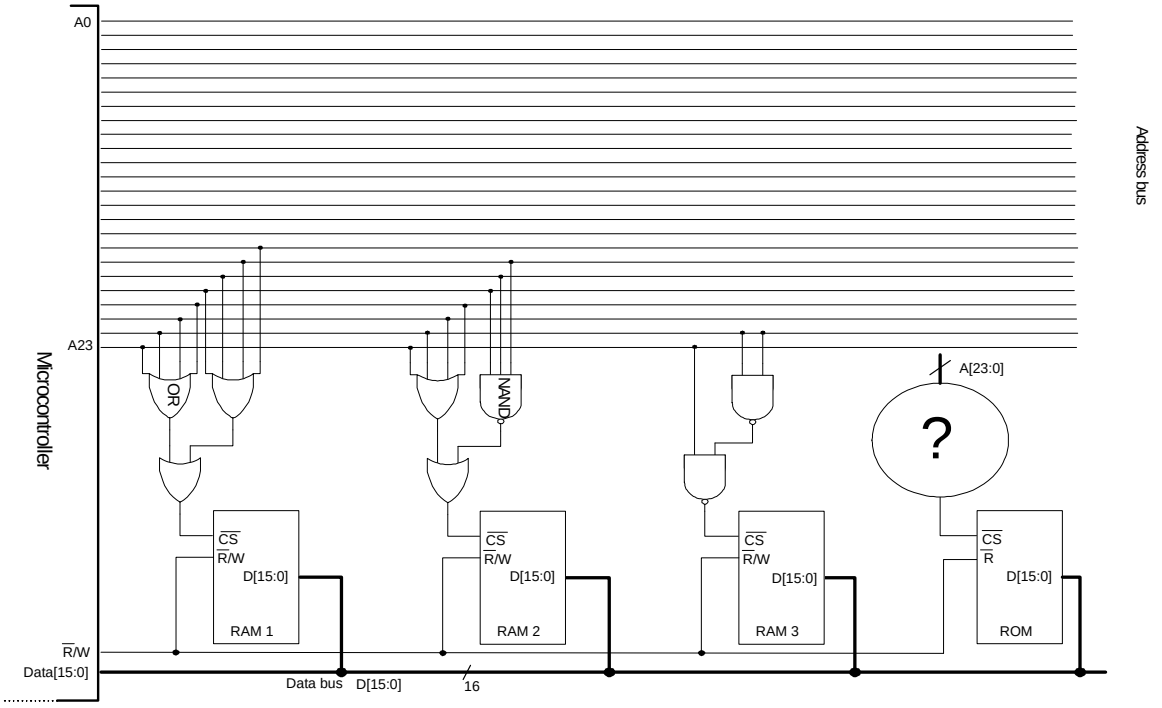
c. Hva kjennetegner superskalareprosessorer?

d. Memory wall (minne veggen) er et begrep som blant annet skaper problemer innen skalering. Hvilke egenskaper ved prosessorer og minne ligger i begrepet?

e. Hvordan påvirkes Instruksjonsnivåparallellitet (Instruction Level parallellism (ILP)) av antall trinn i samlebånd (pipeeline)? Forklar kort.

OPPGAVE 2: DIGITALT LOGISK NIVÅ (25 % (15 % PÅ A, 10 % PÅ B))

I figur 2 er det eksterne bussgrensesnittet for en mikrokontroller vist. Alle enhetene har et aktivt lavt (logisk ”0”) CS (Chip Select)-signal.



Figur 2: Address decoding.

a. i) Hva er adresseområde for RAM 1, RAM 2 og RAM 3?
   ii) Tegn minnekart (memory map) for systemets RAM-brikker.
   iii) Er det overlapp/konflikter i RAM-adresserommet?

b. i) Hva er det maksimale antall addresser som kan utnyttest til ROM-brikken? Angi antall adresser (ord) eller tegn et adresse kart.
   ii) Angi maksimal størrelse på ROM-brikke i bytes.

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

Bruk vedlagte diagram i figur 4, figur 5, figur 6, figur 7, figur 8 og figur 9 for IJVM til å løse oppgavene.

a. Hva brukes registerne MAR og MDR i figur 4 til?

b. Lag mikroinstruksjon(er) for følgende IJVM-operasjon: *OPC = MDR + LV + CPP + 2*. Bruk mikroarkitekturen i figur 4.

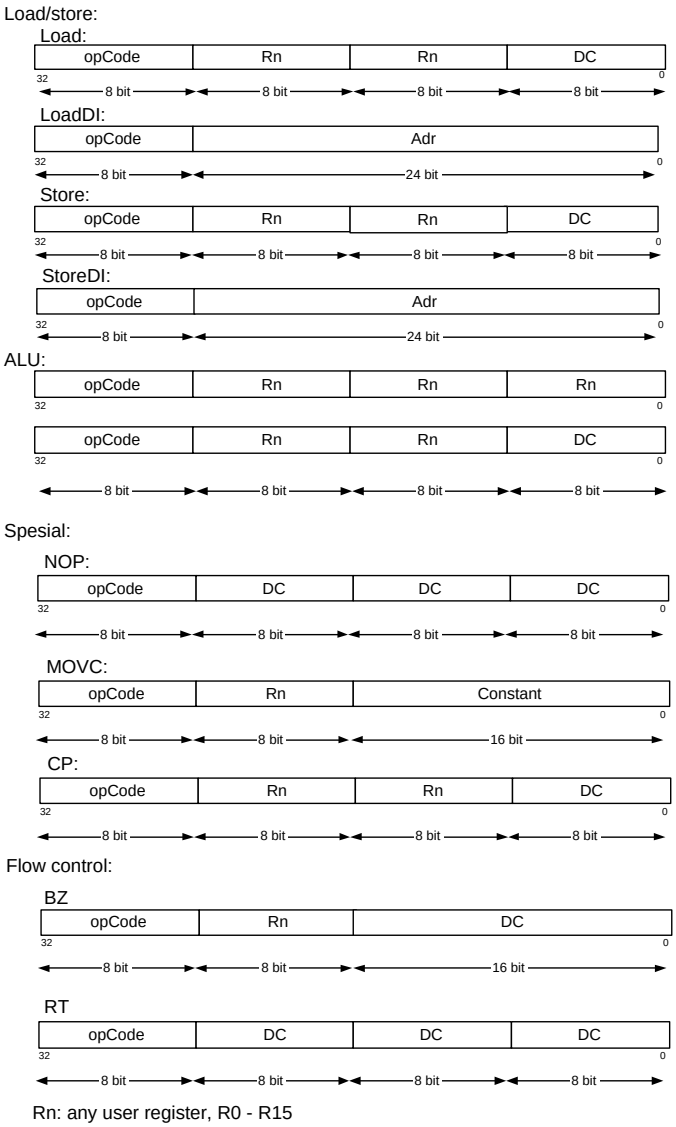
   Du trenger ikke ta hensyn til Addr- og J-feltene. Oppgi korrekt bit-verdi for ALU-, C-, Mem- og B-feltene. – Se figur 5.

c. i) For å få bedre ytelse blir mikroarkitekturen endret til å bruke datapath som er vist i figur 8.Hvordan vil dette påvirke mikroinstruksjonene for instruksjonen: *OPC = MDR + LV + CPP + 2*? Gi eksempel på hvordan den nye mikroarkitekturen kan øke ytelsen.

   ii) For ytterliger øking av ytelse skiftest det til mikroarkitekturen vist i figur 9. Hvilke ytelseforbedrings tiltak er gjort i figur 9?

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

En svært enkel prosessor har to laste- og lagre-instruksjon (load and store instruction), 8 ALU-instruksjoner, noen spesielle instruksjoner som inkluderer NOP-instruksjonen og to flytkontroll-instruksjoner (flow control instructions). Instruksjonsformatet er vist i figur 3. Alle registre og busser er 32-bit. Prosessoren har en Harvard-arkitektur.



Figur 3: Instruction formats.

Instructions set:

LOAD: Load data from memory.

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

LOADDI: Load data from memory.

load Adr Load register R0 from memory location Adr.

STORE: Store data in memory.

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

STOREDI: Store data in memory.

store Adr Store register R0 in memory Adr.

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 Bitwice OR, $Rn = Rn + Rn$. Set Z-flag if result =0.
INV Rn, Rn Bitwice 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, $Rn < -Rn$
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. Target register to the left, i.e. ADD R1, R1 , R3: R1 = R1 + R3.
DC: Don't care.

a. En tenkt kodesnutt leser to verdier fra to minneadresser. Hvordan kan PC (Program Counter) avhenge av innholdet i disse minneadressene for den oppgitte sudokoden (Hvilke mulige verdier kan PC ha etter at koden har kjørt)?

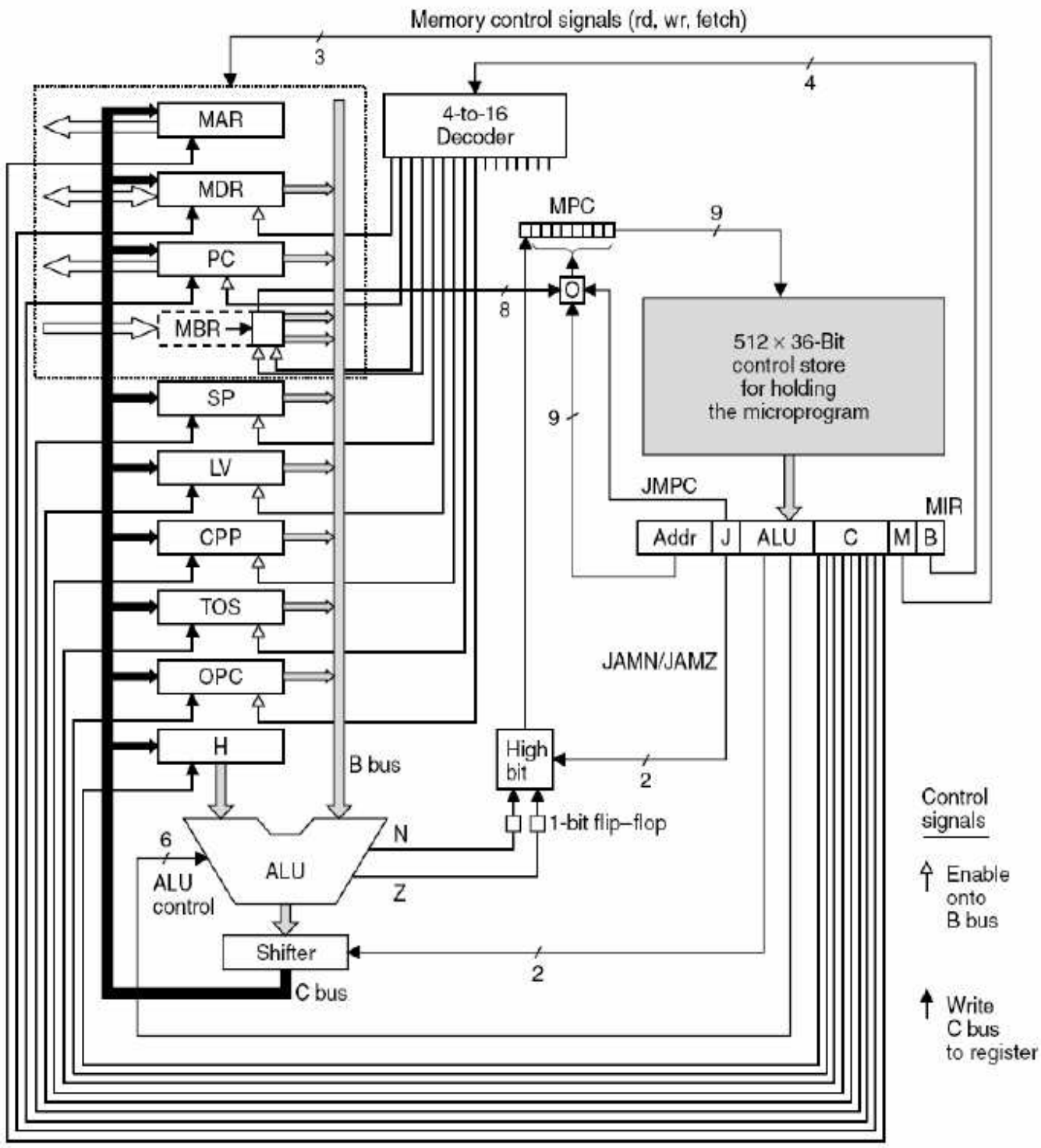
   - LOADDI 0x00 FF FF 00;
   - CP R1, R0;
   - LOADDI 0x00 FF FF 04;
   - OR R0, R0, R1;
   - BZ R0

b. 1) Angi hvilke adresseringsmodi (addressing modes) som brukes i følgende instruksjoner:
   i) LOADDI
   ii) LOAD
   ii) MOVEC

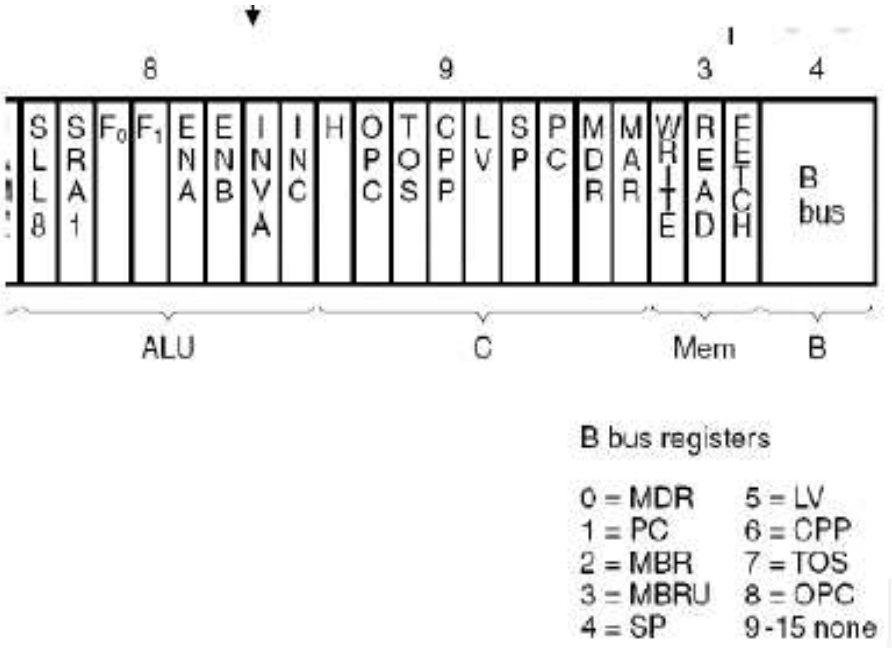
   2) Ut fra gitt instruksjonssett gi eksempel på instruksjonstype (instruction format):
   i) Null adresse instruksjon (Zero-address instruction).
   ii) Tre adresse instruksjon (Three-address instruction).

c. Er det mulig å utvide instruksjonsettet med fleire instruksjonar, f.eks. med shift instruksjonar, utan å endre på ISA? Begrunn svaret.

IJVM appendix



Figur 4: Block diagram (IJVM).



Figur 5: Microinstruction format (IJVM).

Answer Key for The Exam

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

a. Unrer konstruksjon av et datamaskinsystem er det krav til minnesystemet om en gjennomsnittlig minneaksesstid (memory access time) på minimum 4.5ns. To forskjellige løsninger er tilgjengelig.

Minnesystem-A har et nivå med hurtigbuffer (cache) med en aksesstid på 2ns og med et trefforholdstall (hit ratio) på 90 %. Hovedminnet har en aksesstid på 20ns.

Minnesystem-B har et nivå med hurtigbuffer (cache) med en aksesstid på 1ns og med et trefforholdstall (hit ratio) på 60 %. Hovedminnet har en aksesstid på 15ns.

Tilfredstiller noen av de to minnesystemene kravet om 4.5ns gjennomsnittlig minneaksesstid? I så fall hvilket?

Answer: Går ut frå at all minneaksess går via cache:
mean access time = $c + ((1 - h) * m)$;
Minnesystem-A: $2 + (1 - 0.9) * 20) = 4ns$
Minnesystem-B: $1 + (1 - 0.6) * 15) = 7ns$

Viss ein går ut frå at ein kan aksessere minne direkte også får ein noko slikt: mean access time = $h * c + ((1 - h) * m)$;
Minnesystem-A: $2 * 0.9 + (1 - 0.9) * 20) = 3.8ns$
Minnesystem-B: $1 * 0.6 + (1 - 0.6) * 15) = 6.6ns$
Ikkje heilt korrekt men gir rett konklusjon.

b. Ut fra informasjonen i figur 1, er dette en asynkron eller synkron bussoverføring? Begrunn svaret kort.

Answer: Ingen klokke = asynk.

c. Hva kjennetegner superskalareprosessorer?

Answer: Duplisering av utførende enheter i datapath", samlebånd kan då innholde dupliserte einingar, f.eks. duplisere ALU trinn. (Høg ILP).

d. Memory wall (minne veggen) er et begrep som blant annet skaper problemer innen skalering. Hvilke egenskaper ved prosessorer og minne ligger i begrepet?

Answer: Prosessor ytelsen aukar fortare enn minne ytelsen. Prossesor ytelse, klokke og prosesseringskapasitet (throughput), aukar mykje fortare enn minne aksesstid minkar.

e. Hvordan påvirkes Instruksjonsnivåparallellitet (Instruction Level parallellism (ILP)) av antall trinn i samlebånd (pipeeline)? Forklar kort.

Answer: ILP aukar ved auke i samlebandsteg. Fordi antall instruksjoner som samlebandet behandlar aukar.

OPPGAVE 2: DIGITALT LOGISK NIVÅ (25 % (15 % PÅ A, 10 % PÅ B))

I figur 2 er det eksterne bussgrensesnittet for en mikrokontroller vist. Alle enhetene har et aktivt lavt (logisk ”0”) CS (Chip Select)-signal.

a. i) Hva er adresseområde for RAM 1, RAM 2 og RAM 3?
   ii) Tegn minnekart (memory map) for systemets RAM-brikker.
   iii) Er det overlapp/konflikter i RAM-adresserommet?

   Answer: i) RAM 1: 0x00 00 00 - 0x00 FF FF
   RAM 2: 0x0E 00 00 - 0x0F FF FF
   RAM 3: 0x80 00 00 - 0xCF FF FF
   ii) Tegning der alle tre RAM-brikker er med.
   iii) Nei, ser lett ut frå adr. kartet.

b. i) Hva er det maksimale antall addresser som kan utnyttest til ROM-brikken? Angi antall adresser (ord) eller tegn et adresse kart.
   ii) Angi maksimal størrelse på ROM-brikke i bytes.

   Answer: i) Adresse tilgjengelig: 0x01 00 00 - 0x0D FF FF , 0x10 00 00 - 0x7F FF FF og 0xD0 00 00 - 0xFF FF FFF. Eller tegn kart. Eller angi ledige adresser.
   ii) dobbelt av i). Sidan databussen er på 16 bit.

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

Bruk vedlagte diagram i figur 4, figur 5, figur 6, figur 7, figur 8 og figur 9 for IJVM til å løse oppgavene.

a. Hva brukes registerne MAR og MDR i figur 4 til?

Answer: Grensesnitt mot eksterntminne. MAR: peikar til minne adresse i eksternt minne. MDR data som skal skrivast til minne adresse gitt i MAR ved skriveoperasjon, eller innhold i minneadresse peika på av MAR ved leseopperasjon.

b. Lag mikroinstruksjon(er) for følgende IJVM-operasjon: *OPC = MDR + LV + CPP + 2*. Bruk mikroarkitekturen i figur 4.

Du trenger ikke ta hensyn til Addr- og J-feltene. Oppgi korrekt bit-verdi for ALU-, C-, Mem- og B-feltene. – Se figur 5.

Answer: Ein mulig løysing:
1: ALU: B-function, B-bus: MDR, C-bus: H [010100][0 (0000)][100000000]
2: ALU: A + B + 1, B-Bus: LV, C-bus: H [111101][5 (0101)][100000000]
3: ALU: A + B + 1, B-bus: CPP, C-bus: OPC [111101][6 (0110)][010000000]

Alle løysingar som gir *OPC = MDR + LV + CPP + 2* er like rett.

c. i) For å få bedre ytelse blir mikroarkitekturen endret til å bruke datapath som er vist i figur 8.Hvordan vil dette påvirke mikroinstruksjonene for instruksjonen: *OPC = MDR + LV + CPP + 2*? Gi eksempel på hvordan den nye mikroarkitekturen kan øke ytelsen.

ii) For ytterliger øking av ytelse skiftest det til mikroarkitekturen vist i figur 9. Hvilke ytelseforbedrings tiltak er gjort i figur 9?

Answer: i):
Kan nå utnytte den nye bussen for å slippe å bruke H-registeret for alle to-operand ALU-funksjoner. F.eks. for *OPC = MDR + LV + CPP + 2*:
1 : ALU: A + B + 1, A-Bus: MDR, B-Bus: LV, C-bus: OPC
2 : ALU: A + B + 1, A-Bus: OPC, B-Bus: CPP, C-bus: OPC

ii):
Pipelining som gir økt ILP samt muligheit for høgare klokkefrekvens. (det er også lagt til ein IFU men ikkje nødvendigt for korrekt svar).

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

En svært enkel prosessor har to laste- og lagre-instruksjon (load and store instruction), 8 ALU-instruksjoner, noen spesielle instruksjoner som inkluderer NOP-instruksjonen og to flytkontroll-instruksjoner (flow control instructions). Instruksjonsformatet er vist i figur 3. Alle registre og busser er 32-bit. Prosessoren har en Harvard-arkitektur.

Instructions set:

LOAD: Load data from memory.

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

LOADDI: Load data from memory.

load Adr Load register R0 from memory location Adr.

STORE: Store data in memory.

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

STOREDI: Store data in memory.

store Adr Store register R0 in memory Adr.

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 Bitwice OR, $Rn = Rn + Rn$. Set Z-flag if result =0.
INV Rn, Rn Bitwice 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, $Rn < -Rn$
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. Target register to the left, i.e. ADD R1, R1 , R3: R1 = R1 + R3.
DC: Don't care.

a. En tenkt kodesnutt leser to verdier fra to minneadresser. Hvordan kan PC (Program Counter) avhenge av innholdet i disse minneadressene for den oppgitte sudokoden (Hvilke mulige verdier kan PC ha etter at koden har kjørt)?

- LOADDI 0x00 FF FF 00;
- CP R1, R0;
- LOADDI 0x00 FF FF 04;
- OR R0, R0, R1;
- BZ R0

Answer: PC vil være = 0 viss begge minne lokasjonene 0x00 FF FF 00 og 0x00 FF FF 04 er 0 (resultatet av OR = 0). Ellers vil PC peke på neste instruksjon, i.e. Pc = PC + 1. Gir litt slakk her viss dei skjønnar kva som skjer.

b. 1) Angi hvilke adresseringsmodi (addressing modes) som brukes i følgende instruksjoner:
i) LOADDI
ii) LOAD
ii) MOVEC

2) Ut fra gitt instruksjonssett gi eksempel på instruksjonstype (instruction format):
i) Null adresse instruksjon (Zero-address instruction).
ii) Tre adresse instruksjon (Three-address instruction).

Answer: 1)
i) Direct Addressing.
ii) Register Addresing.
ii) Imidiate Addressing
. 2)
i) f.eks. NOP og/eller RT er rett svar
ii) Dei flestr ALU, alle med oppCode Rn, Rn, Rn

c. Er det mulig å utvide instruksjonsettet med fleire instruksjonar, f.eks. med shift instruksjonar, utan å endre på ISA? Begrunn svaret.

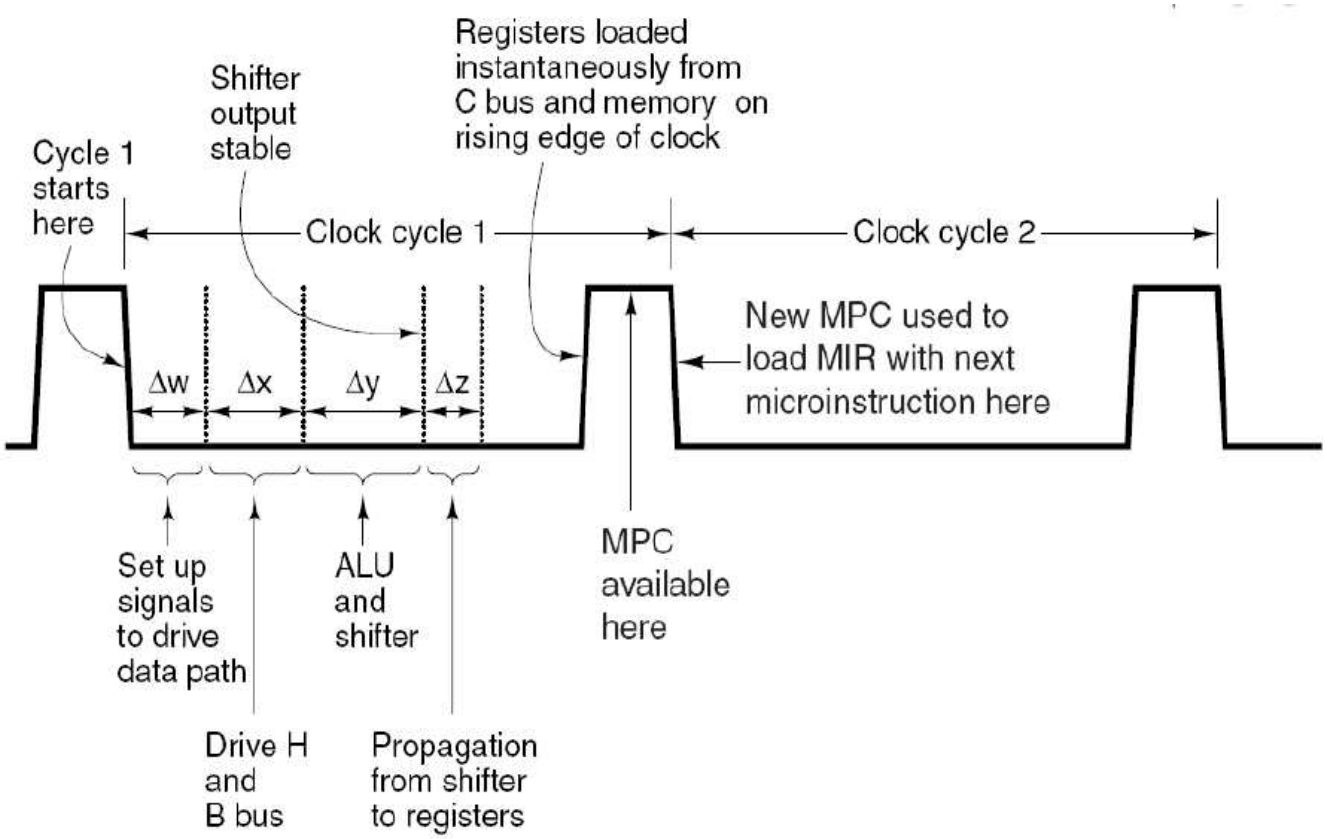
Answer: Nei, viss ein endrar instruksjonssette endrast ISA.

IJVM appendix

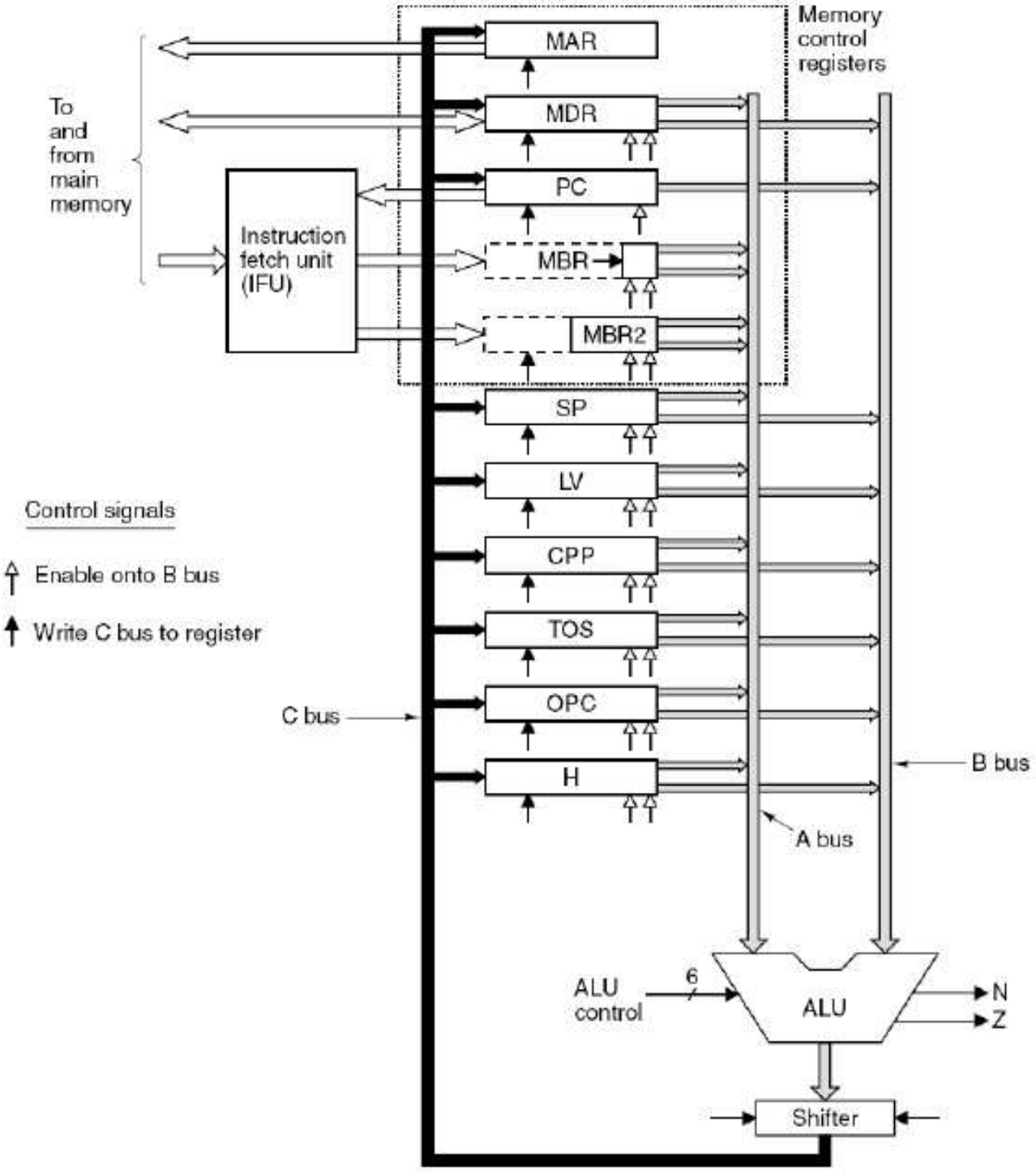
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

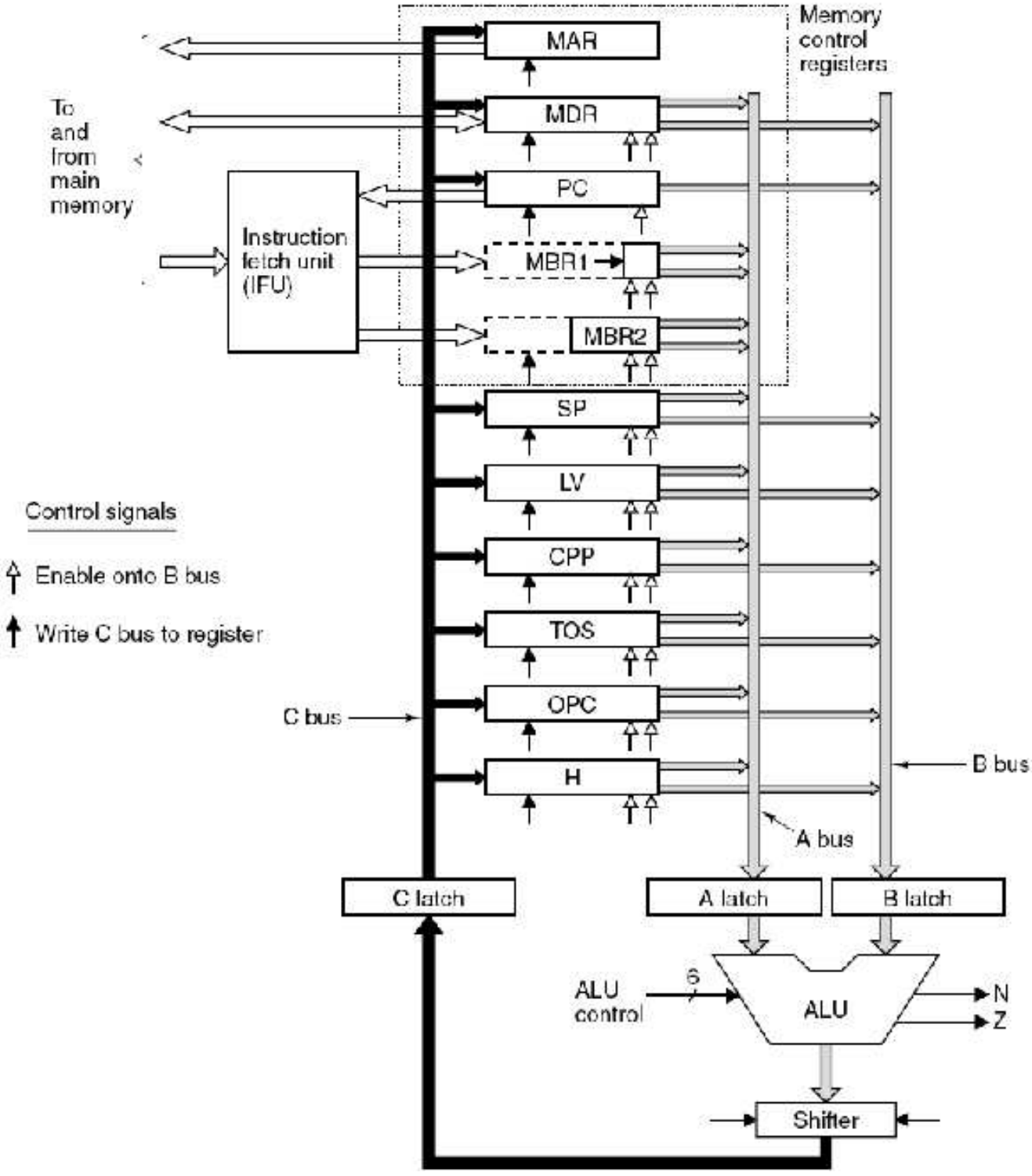
Figur 6: ALU functions (IJVM).



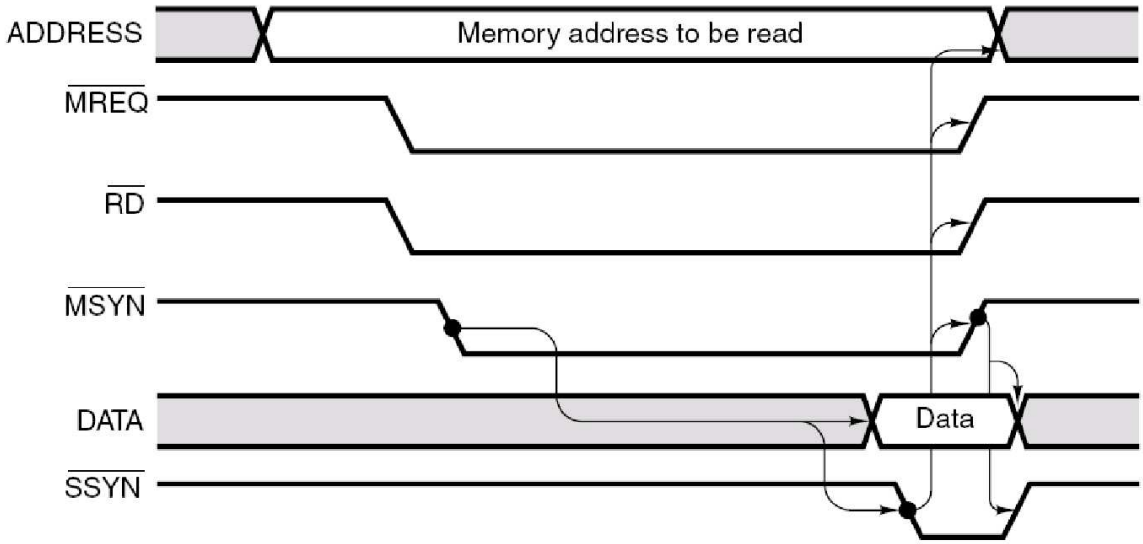
Figur 7: Timing diagram (IJVM).



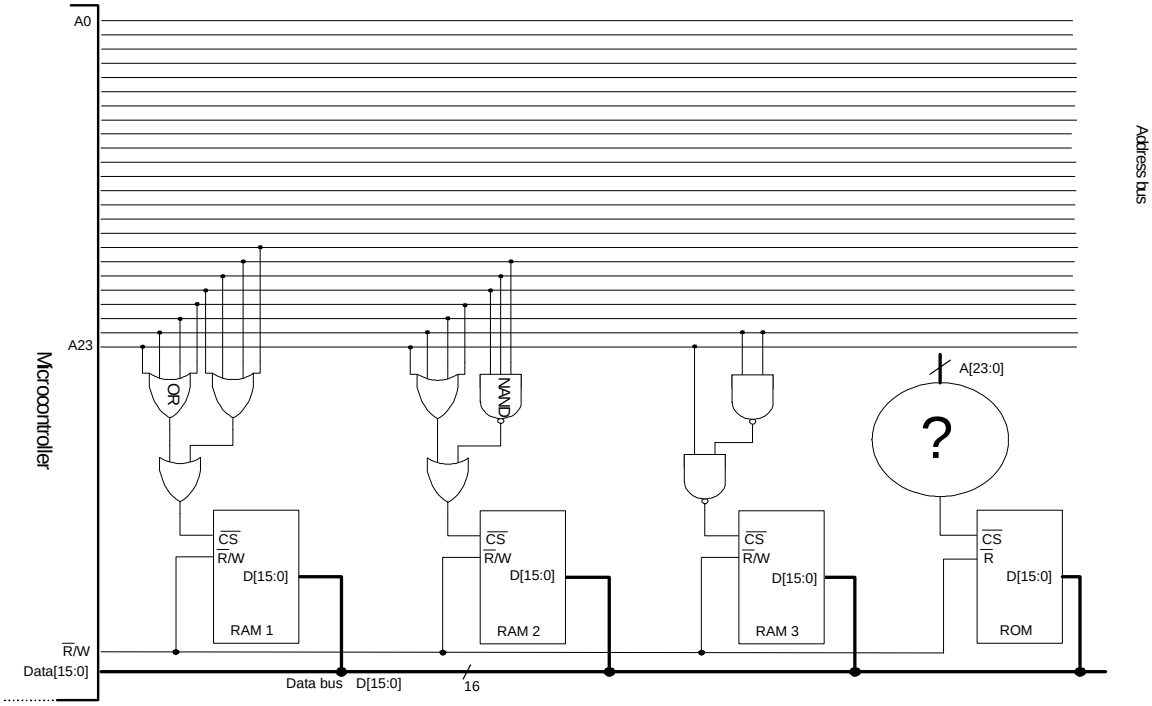
Figur 8: Alternative microarchitecture I.



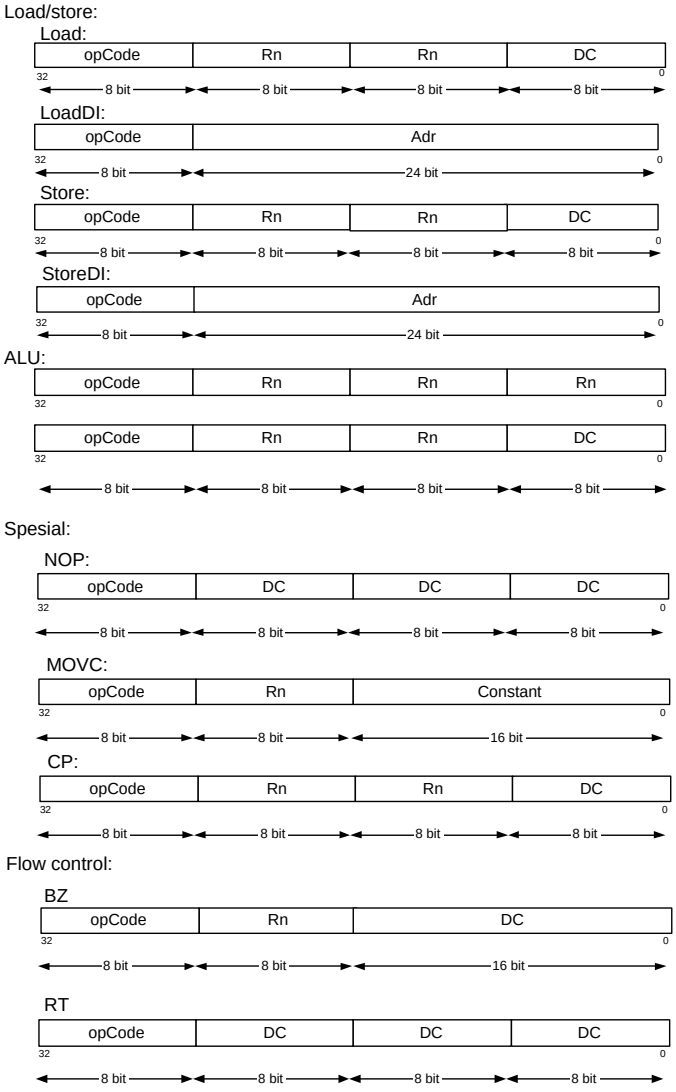
Figur 9: Alternative microarchitecture II.



Figur 10: Address decoding.

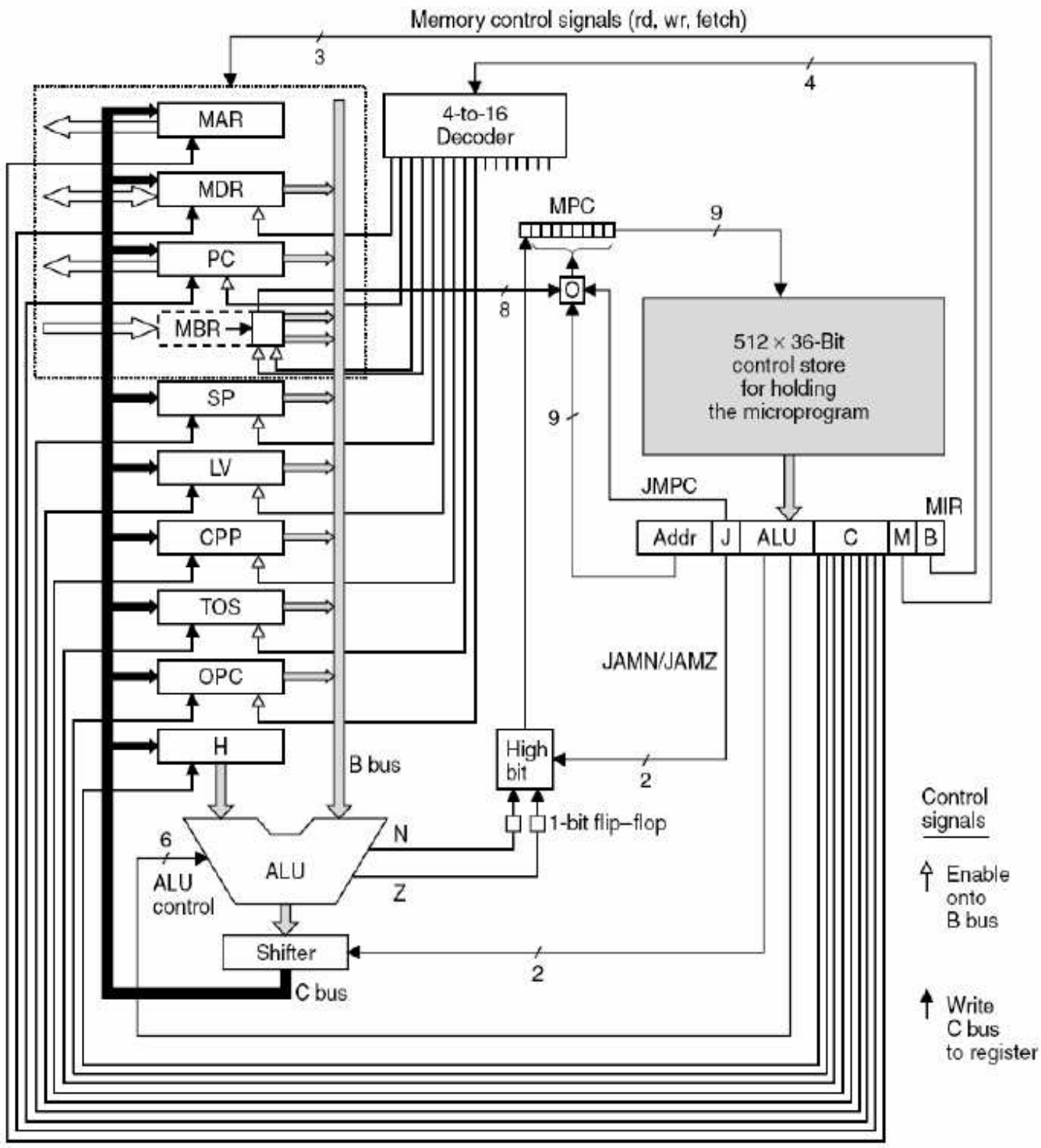


Figur 11: Address decoding.

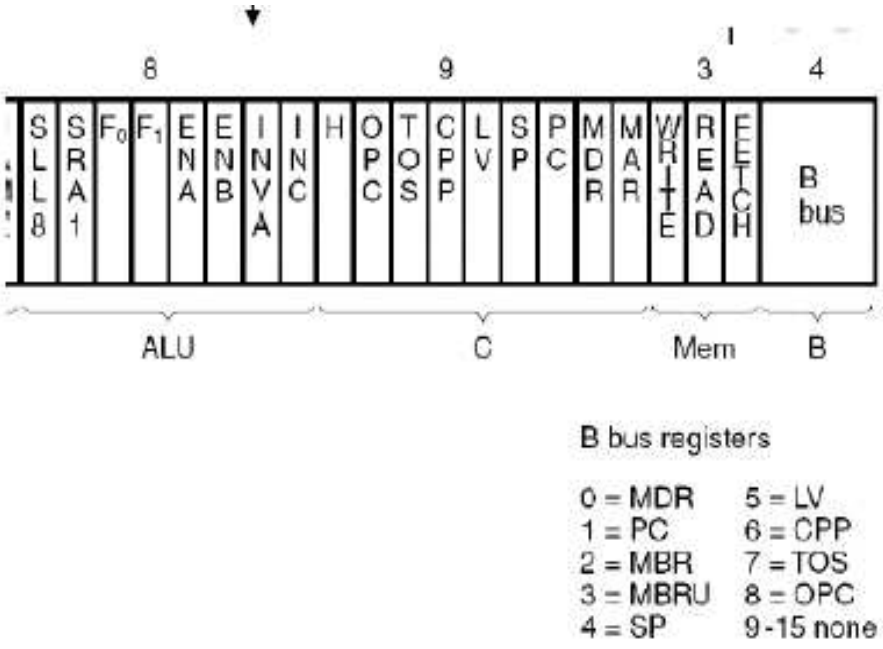


Rn: any user register, R0 - R15
DC: Don't care. Adr: any data memory location within range.

Figur 12: Instruction formats.



Figur 13: Block diagram (IJVM).

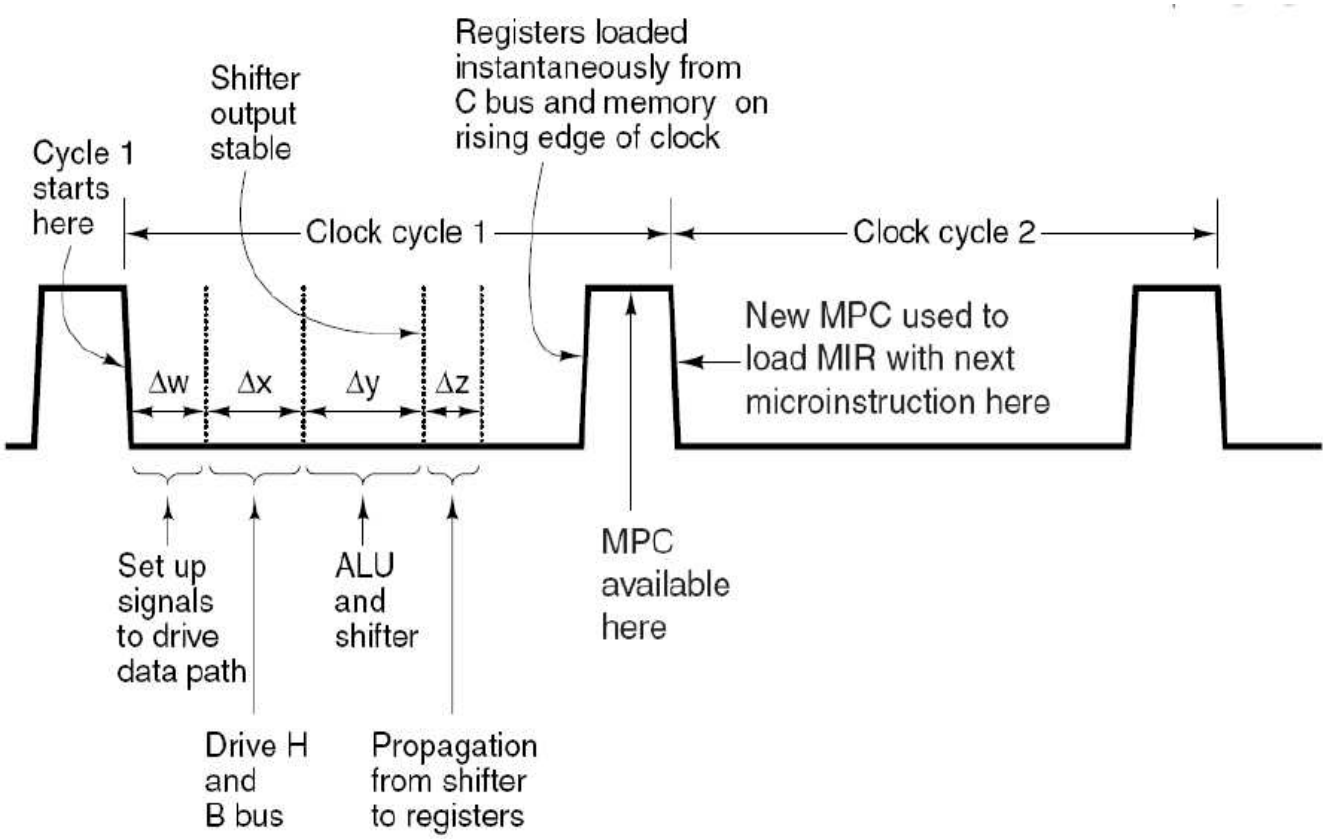


Figur 14: Microinstruction format (IJVM).

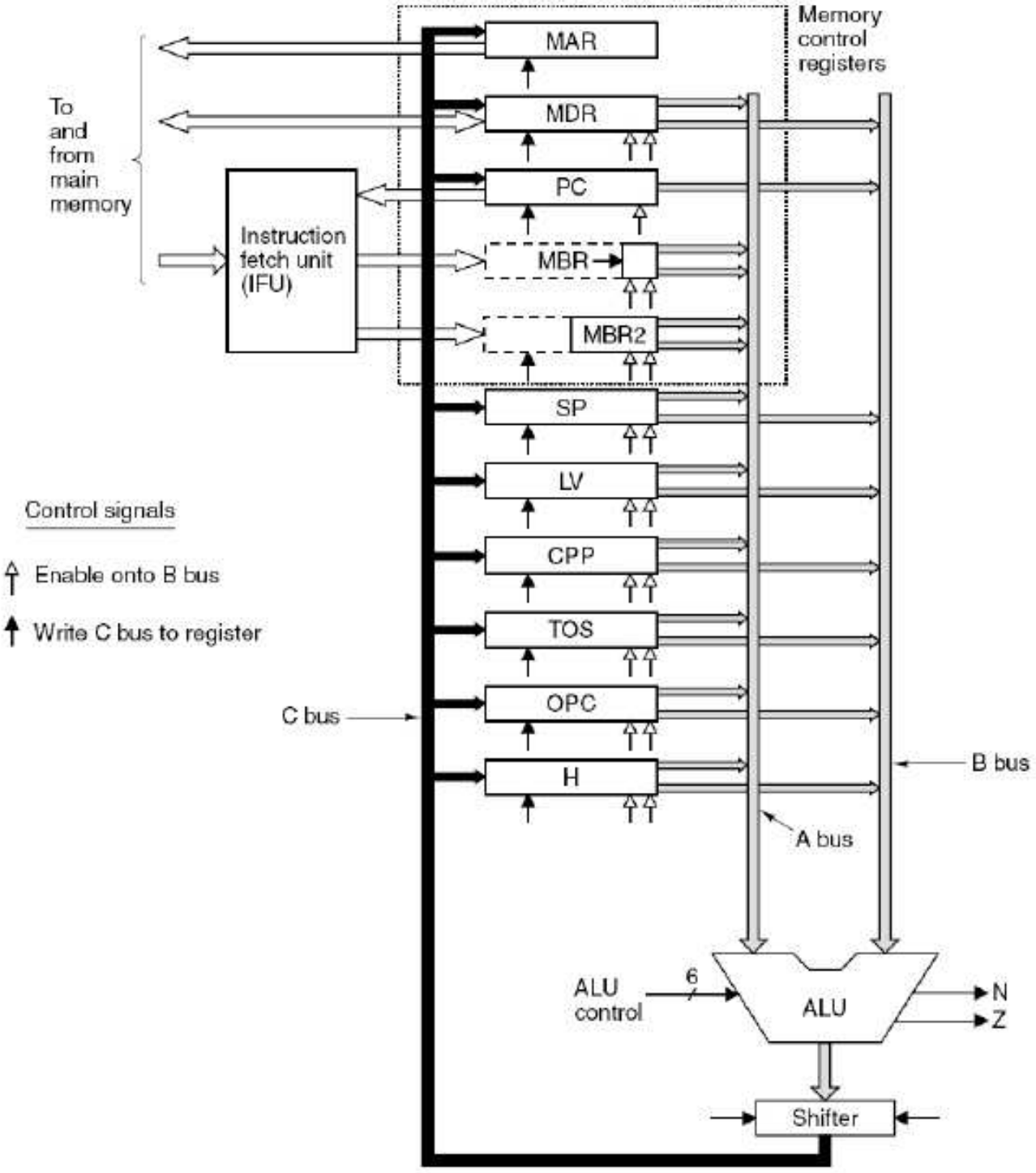
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

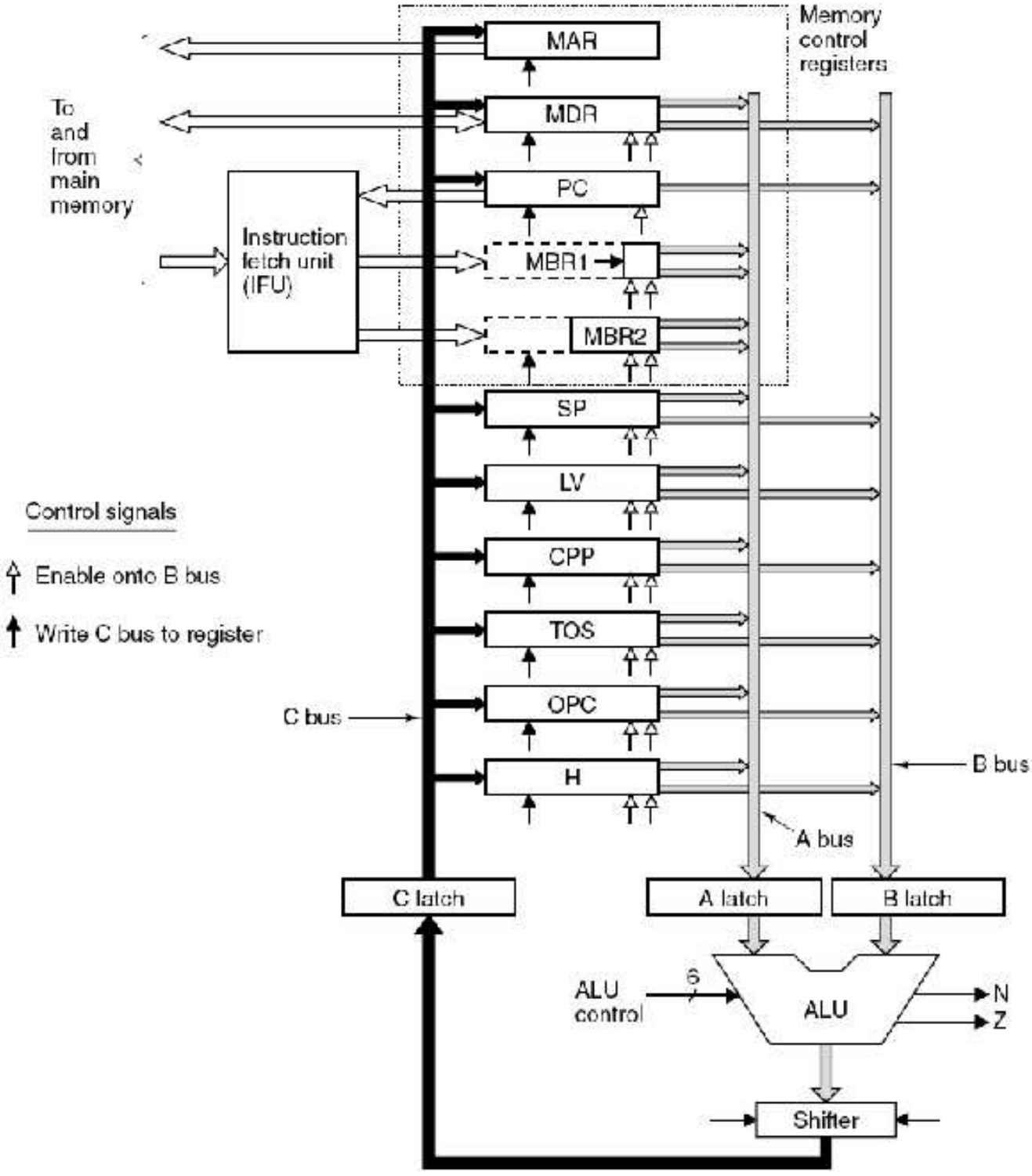
Figur 15: ALU functions (IJVM).



Figur 16: Timing diagram (IJVM).



Figur 17: Alternative microarchitecture I.



Figur 18: Alternative microarchitecture II.