



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 %)

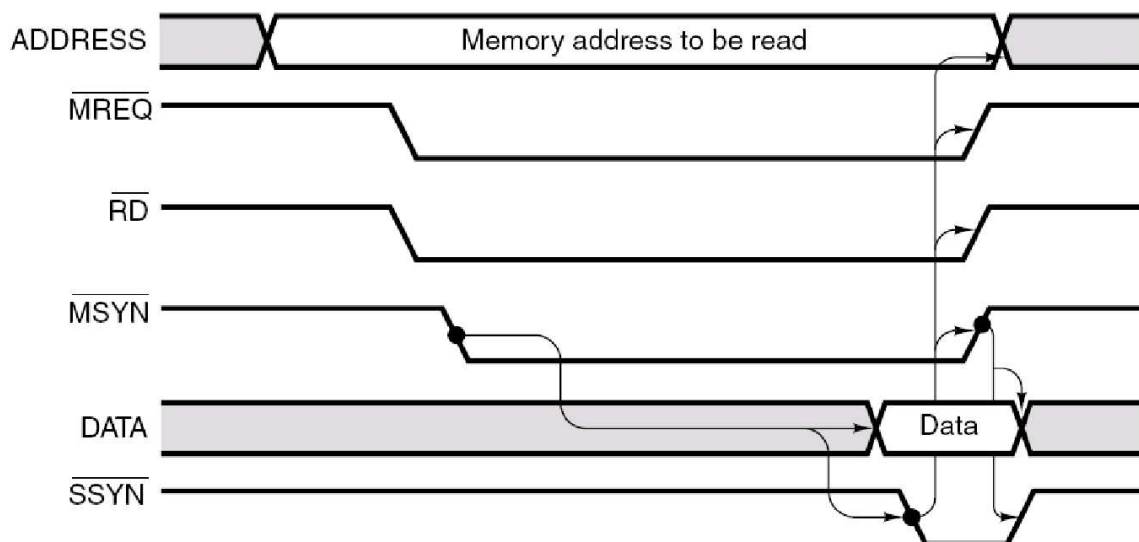
- a. Under konstruksjon av et datamaskinsystem er det krav til minnesystemet om en gjennomsnittlig minneaksessetid (memory access time) på minimum 4.5ns. To forskjellige løsninger er tilgjengelig.

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

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

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

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

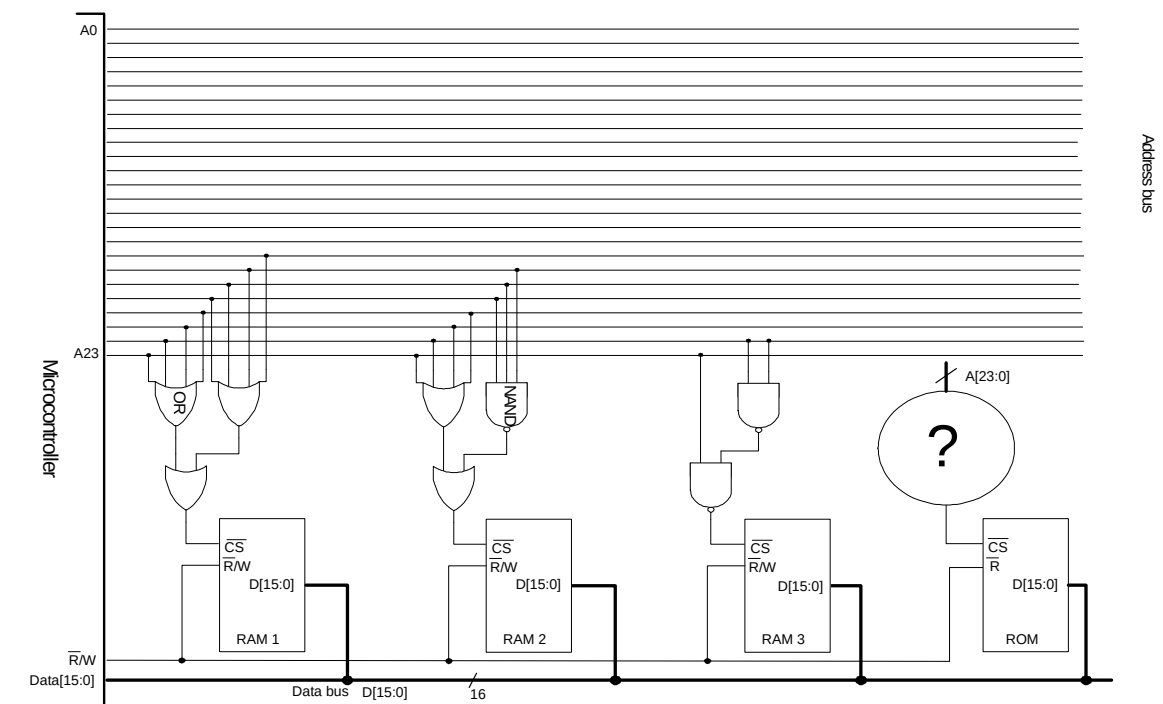


Figur 1: Address decoding.

- 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 Instruksjonsnivåparallelitet (Instruction Level parallelism (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 adresser som kan utnyttast 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))

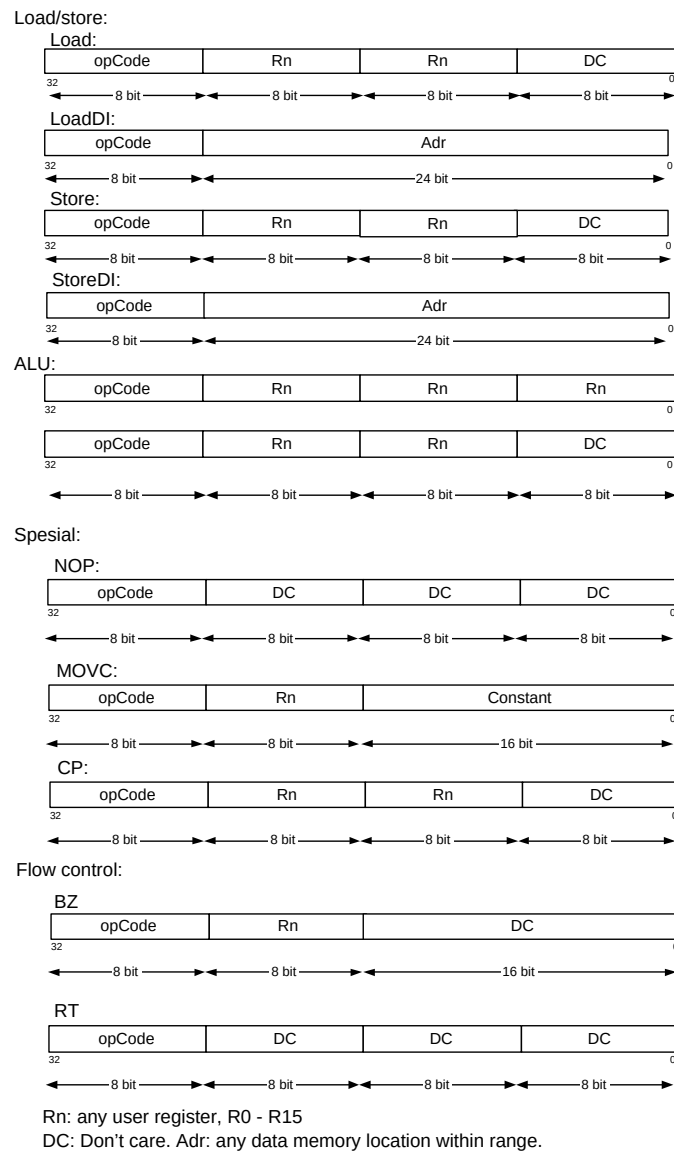
Bruk vedlagte diagram i figur 4, figur 5, figur 6, figur 7, figur 8 og figur 9 for IJVM til å løse oppgavene.

- 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 ytterligere øking av ytelse skiftes 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 Bitwise OR, $Rn = Rn + Rn$. 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, $Rn < -Rn$

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, $PC = Rn$.

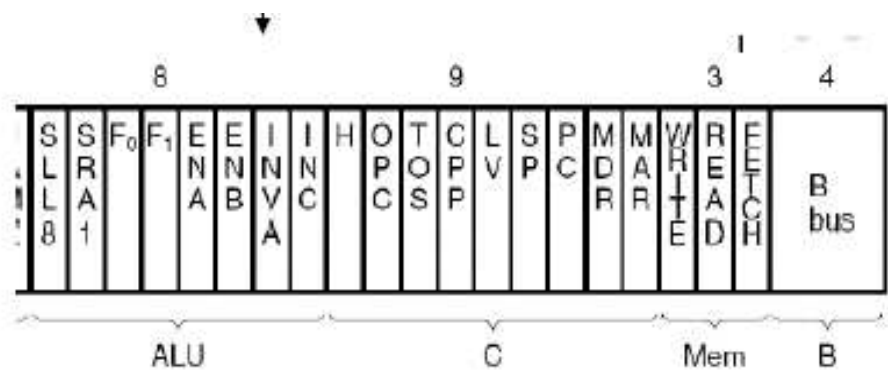
RT Return, return from branch.

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 flere instruksjoner, f.eks. med shift instruksjoner, utan å endre på ISA? Begrunn svaret.

IJVM appendix



B bus registers

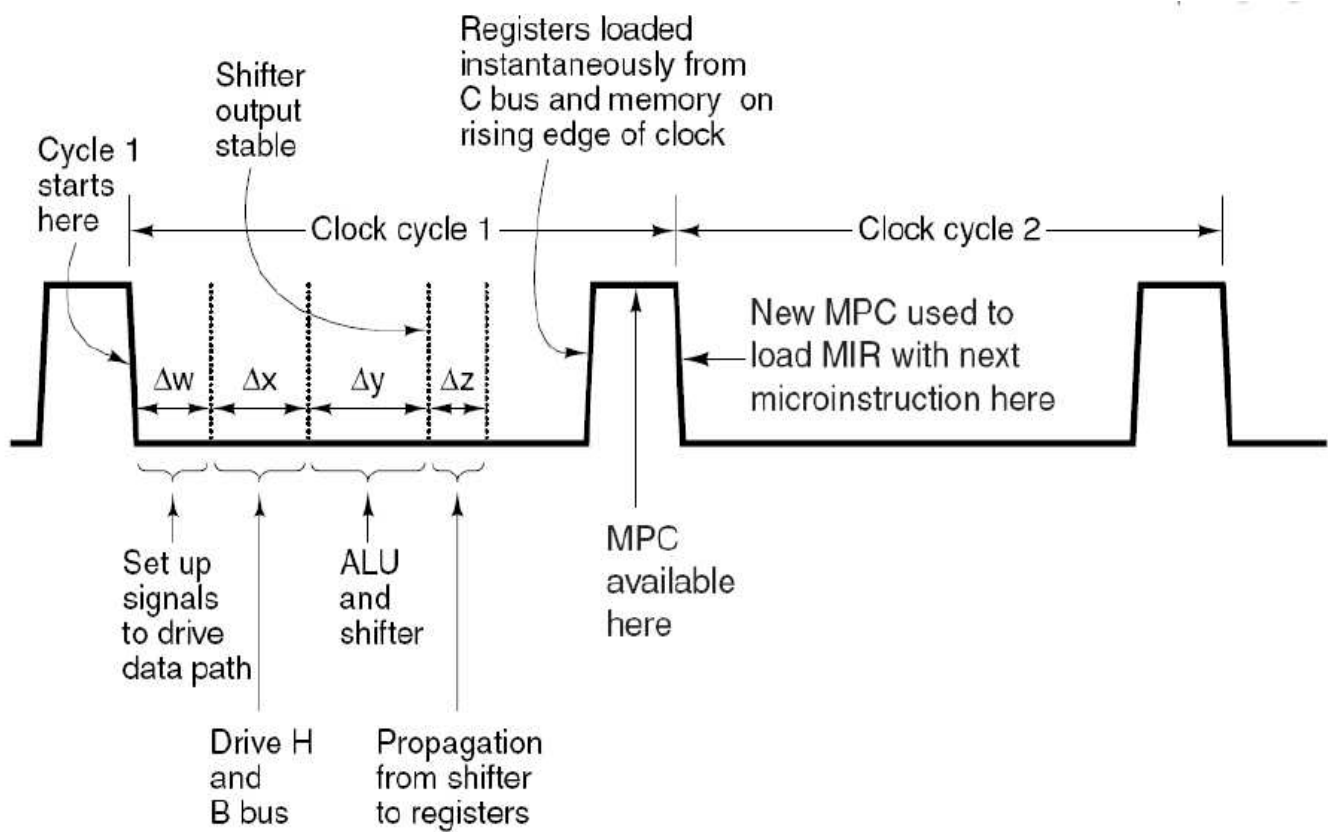
0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Figur 5: 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 6: ALU functions (IJVM).



Figur 7: Timing diagram (IJVM).

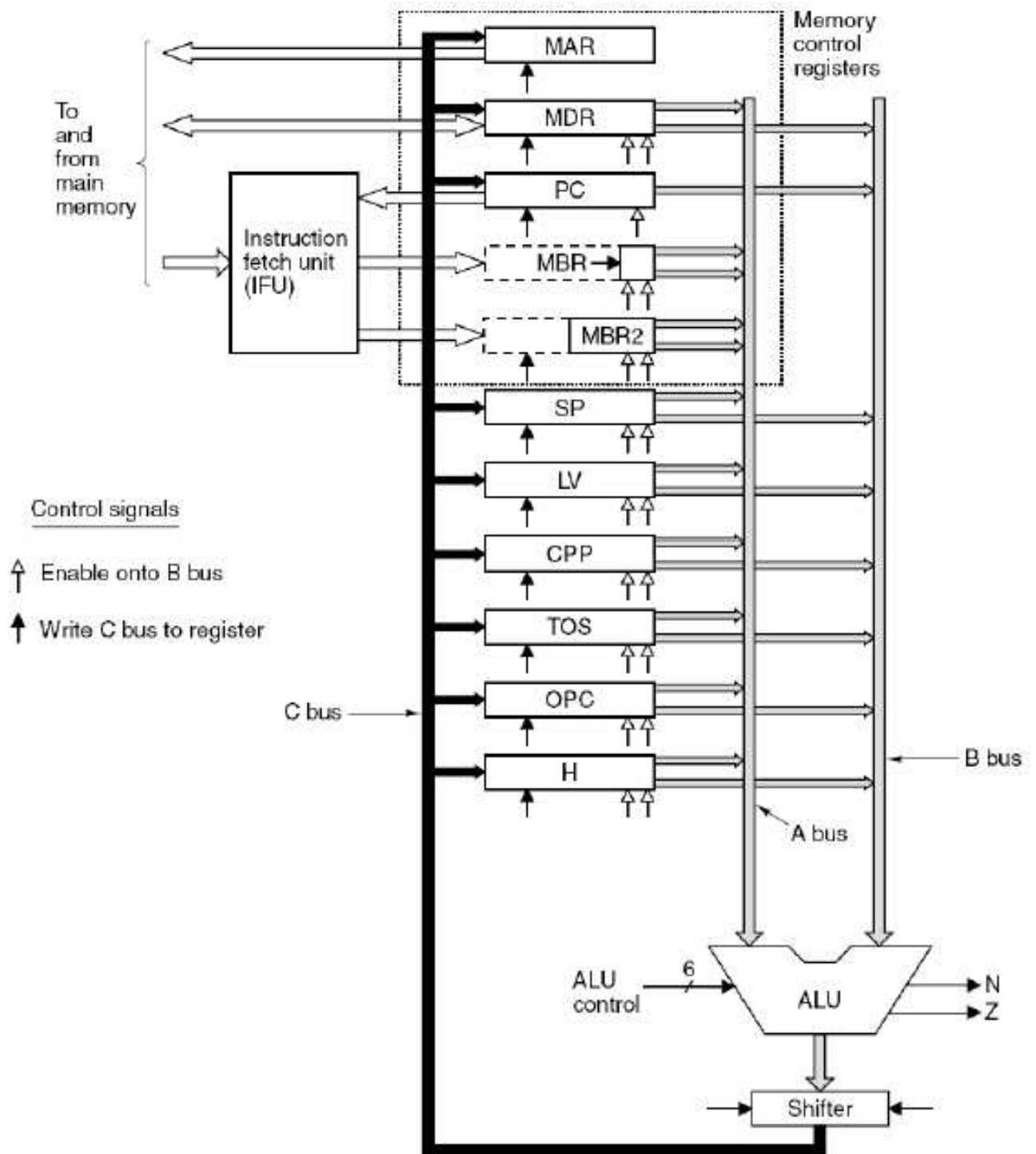
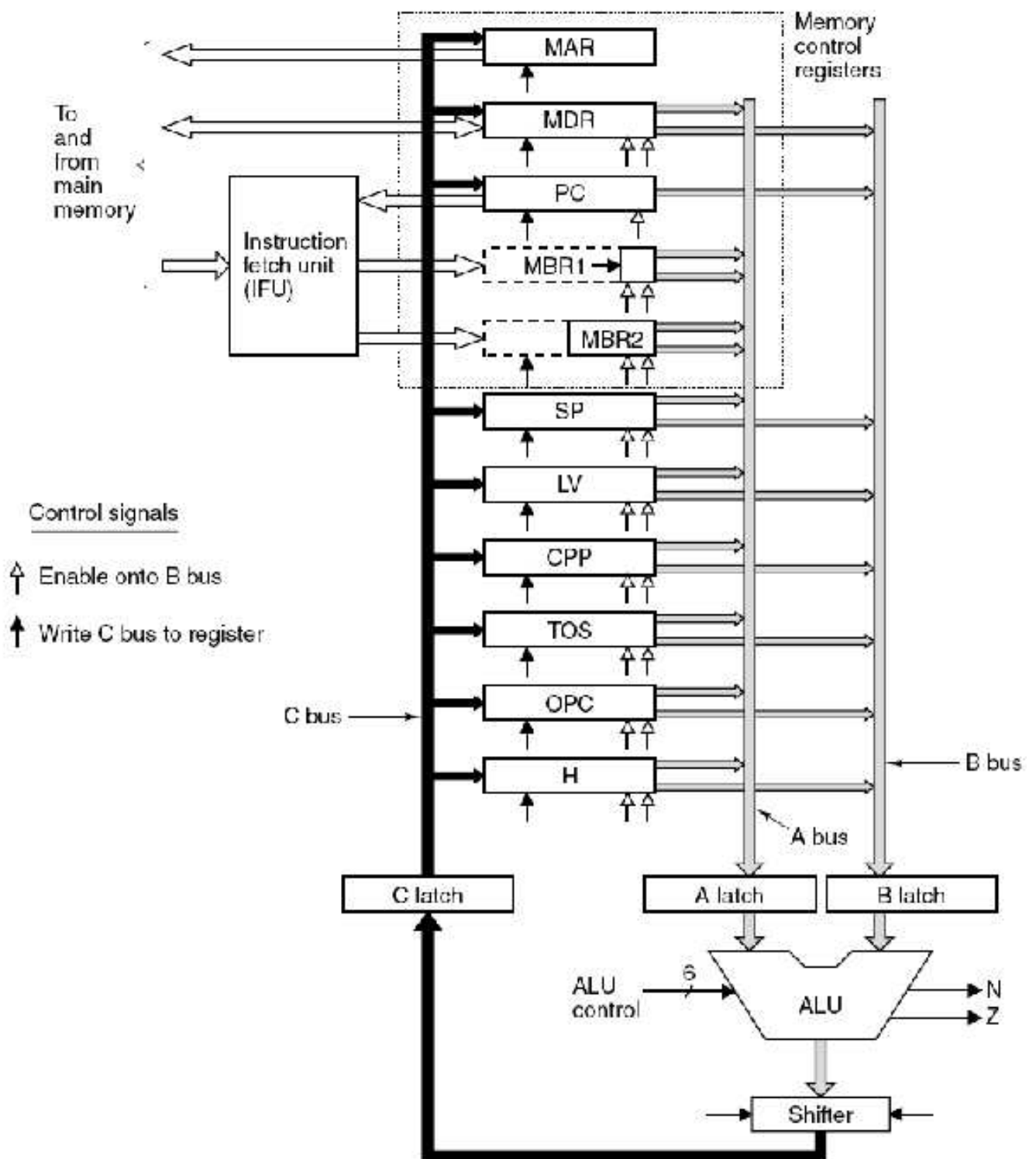


Figure 8: Alternative microarchitecture I.



Figur 9: Alternative microarchitecture II.