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

TDT4160
DATAMASKINER GRUNNKURS
EKSAMEN
17. DESEMBER, 2012, 09:00–13:00

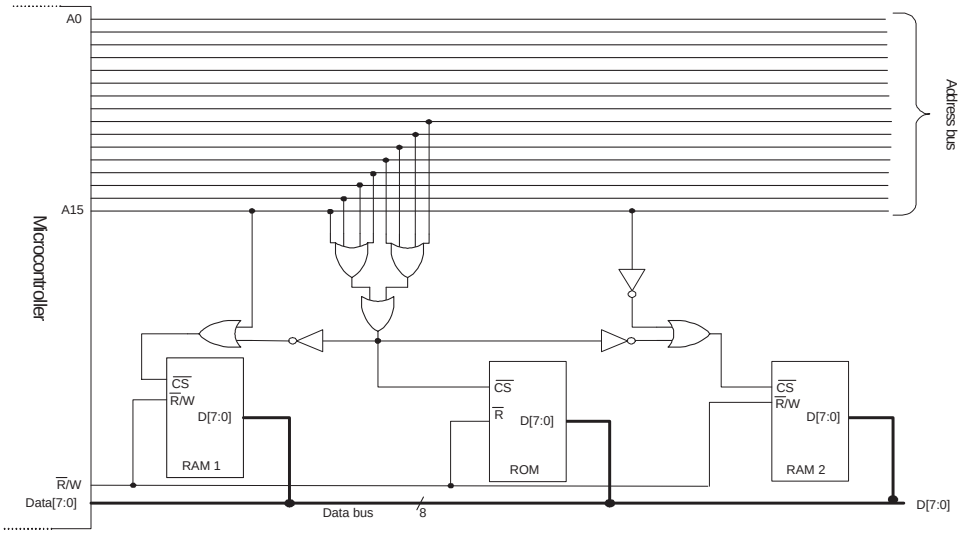
Kontakt under eksamen:
Gunnar Tufte 73590356 / 97402478

Tillatte hjelpemidler:
D.
Ingen trykte eller handskrivne hjelpemiddel er tillete.
Enkel godkjent kalkulator er tillete.

Målform:
Nynorsk

OPPGÅVE 1: DIGITAL LOGISK NIVÅ (25% (5% ON A, 10% ON B AND C))

I figur **??** er det eksterne bussgrensesnittet for ein mikrokontroller vist. Det er brukt to RAM-brikker på 32kB og ein ROM-brikke på 1kB. Adressedekodingen er bygd opp av OR-portar og inverterportar. Alle einingane har eit aktivt lågt (logisk ”0”) CS (Chip Select)-signal.

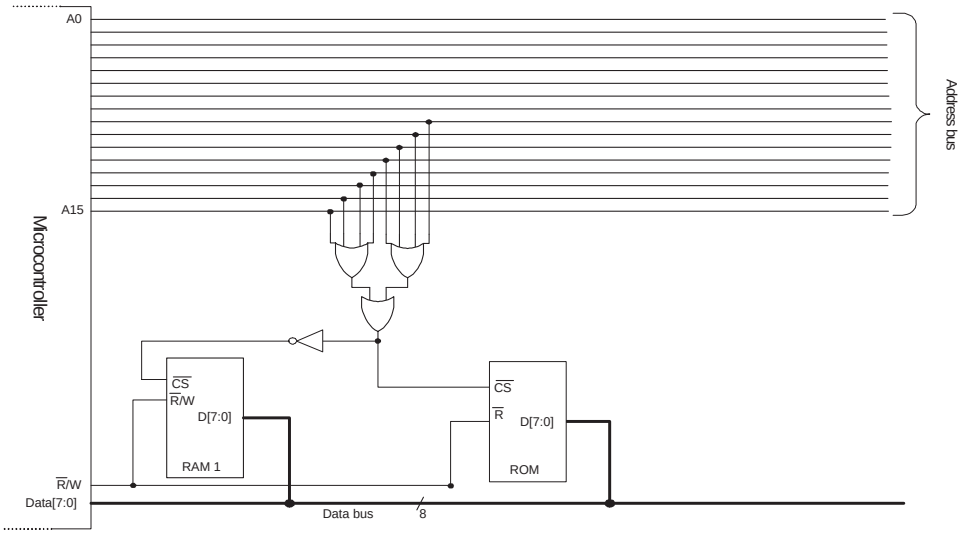


Figur 1: Address decoding.

a. Kva er det maksimale eksterne minneområdet mikrokontrolleren kan adressere?

b. Kva er adresseområdet for RAM 1-, RAM 2- og ROM-brikken? Teikn minnekart for systemet.

c.



Figur 2: New address decoding.

d. For å få ned produksjonskostnaden for systemet, er det foreslått eit nytt maskinvaredesign, sjå figur **??**. I det nye designet er dei to RAM-brikkane erstatta av ein RAM-brikke som er dobbelt så stor.

   i) Er minnekartet for systema i figur **??** og figur **??** identiske?

   ii) Er det mogleg å behalde programvare utvikla for systemet i figur **??** utan endringar på det nye maskinvaredesignet? Forklar.

OPPGÅVE 2: MIKROARKITEKTUR OG MIKROINSTRUKSJONAR (15%)

Bruk vedlagte diagram i figur **??**, figur **??**, figur **??** og figur **??** for IJVM til å løyse oppgåvene.

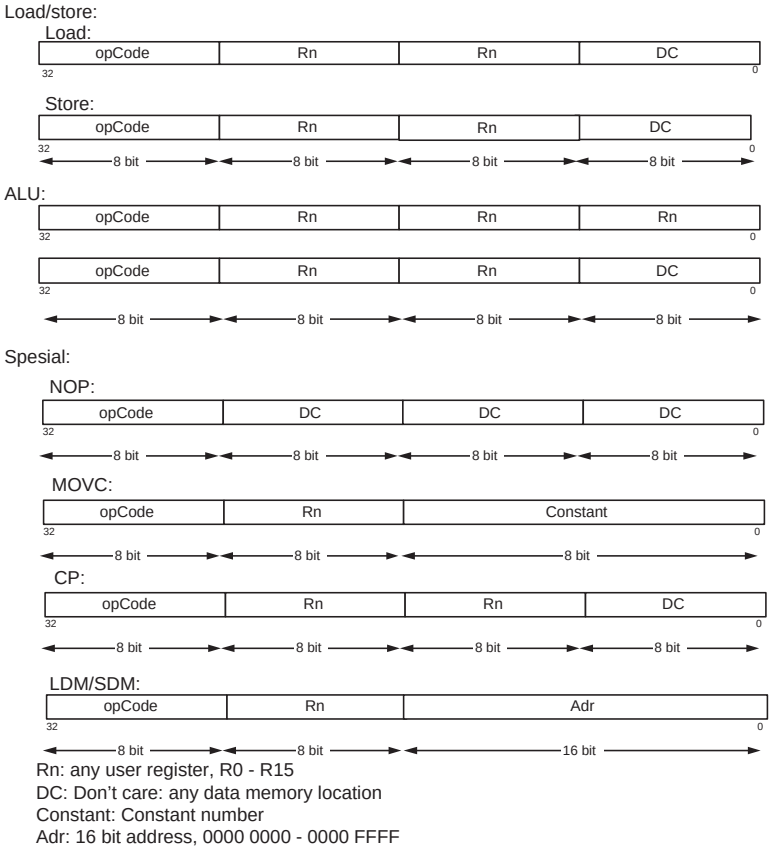
(i) Komponenten 4-to-16 Decoder i figur **??** er brukt til å kontrollere kva som skal liggje på B-bus. For C-Busen er separate bit brukt til å kontrollere kvart register. Kvifor?

(ii) Lag mikroinstruksjon(ar) for følgjande IJVM-operasjon: *OPC = TOS - LV + 1*.

Sjå vekk frå Addr- og J-felta i mikroinstruksjonsformatet. Angi korrekte bit for ALU, C, Mem og B gitt i figur **??**.

OPPGÅVE 3: INSTRUKSJONSSETT ARKITEKTUR(ISA)(25%)

Ein særs enkel prosessor har ein ”load”, ein ”store”, 8 ALU-instruksjonar og nokre spesial-instruksjonar, inkludert NOP-instruksjonen. Instruksjonsformatet for instruksjonane er vist i figur **??**. Alle register og bussar er 32-bit. Prosessoren er av typen "Harvard architecture".



Figur 3: Instruksjonsformat.

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, $Rn = Rn + Rn$.
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, $Rn = Rn - 1$.
MUL Rn, Rn, Rn Multiplication, $Rn = Rn * Rn$.
CMP, Rn, Rn Compare, Z-flag = 1 if $Rn = Rn$, else Z-flag = 0.

Special: Misc.

CP Rn, Rn Copy, $Rn < -Rn$.
NOP Waste of time, 1 clk cycle.
MOVC Rn, constant Put a constant in register, $Rn = C$.
LDM Rn, Adr Load Rn with content of memory Address.
SDM Rn, Adr Store content of Rn in memory Address.

Rn: Any user register.
DC: Don't care.
C: Constant.
Adr: 16 bit address space, 0000 0000 - 0000 FFFF.

(i) Basert på tilgjengeleg informasjon kan ein ikkje definera denne prosessoren til å være generell.

   i) Kvifor?

   ii) Kva må til for at prosessoren skal være generell? Gi eit døme.

(ii) NOP-instruksjonen er ei 0-adresseinstruksjon (Zero-address instruction). Kva andre adresseringar (addressing) er brukt i instruksjonssettet?

(iii) MOVC-instruksjonen flytter ein konstant verdi inn i eit register.

   i) Kva adresseringsmodus (addressing mode) er brukt i MOVC-instruksjonen?

   ii) Når kan verdien til konstanten definerast?

(iv) Er det mogleg å bestemme om denne prosessoren nyttar samleband (pipelining) utfrå tilgjengeleg informasjon? Forklar kvifor / kvifor ikkje.

(v) Kva er det totale minneområdet prosessoren kan adressera?

OPPGÅVE 4: DATAMASKINER OG YTING(25% (10% ON A; 15% ON B))

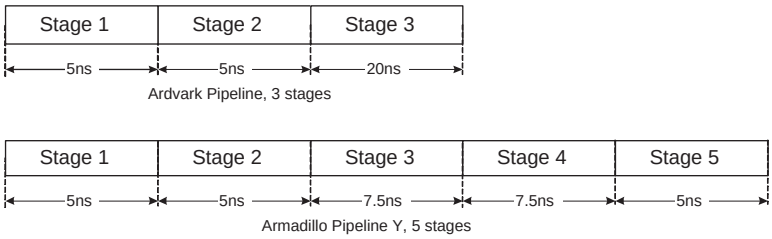
(i) Figure **??** and Figure **??** i appendix viser ulike versjonar av IJVM-mikroarkitekturen. Den orginale mikroarkitekturen er vist i figur **??**.

i) Vil endringen i mikroarkitektur frå figur **??** til figur **??** kreve endringar i MIR? Forklar.

ii) I figur **??** er samleband innført. Er superscalarity ein eigenskap som er tilstades i mikroarkitekturen vist i figur **??**?

iii) Mikroarkitekturane vist i figur **??** og figur **??** krever også endringar i mikroinstruksjonane. Vil ISA-nivået bli påverka av at ein endrar korleis instruksjonar vert utført på mikroarkitekturnivå?

(ii) I ein tenkt prosessorfamilie, iBeast, eksisterar det to versjonar: iAardvark og iArmadillo. iAardvark er eit trestegs samlebanddesign. iArmadillo er eit femstegs samlebanddesign. Figur **??** viser dei to samlebanda med trinnforsinking.



Figur 4: Samlebandtidsbruk for iArmadillo- og iAardvark-versionane av iBeast.

i) Estimer maksimal klokkefrekvens for iArmadillo og iAardvark.

ii) Estimer klokkefrekvens for ein iBeast-prosessorversjon utan samleband.

iii) Prosessorfamilien er produsert i ein 22nm-prosess. Korleis vil mest truleg ein nedskalering til ein 18nm-produksjonsprosess påverke trinnforsinkelsen?

iv) Lat oss si at du vil lage ein Chip Multi Processor (CMP)-version av iBeast-prosessoren. Kva prosessorkjerne, iArmadillo eller iAardvark, vil være det beste valget viss høg ILP er målet? Forklar.

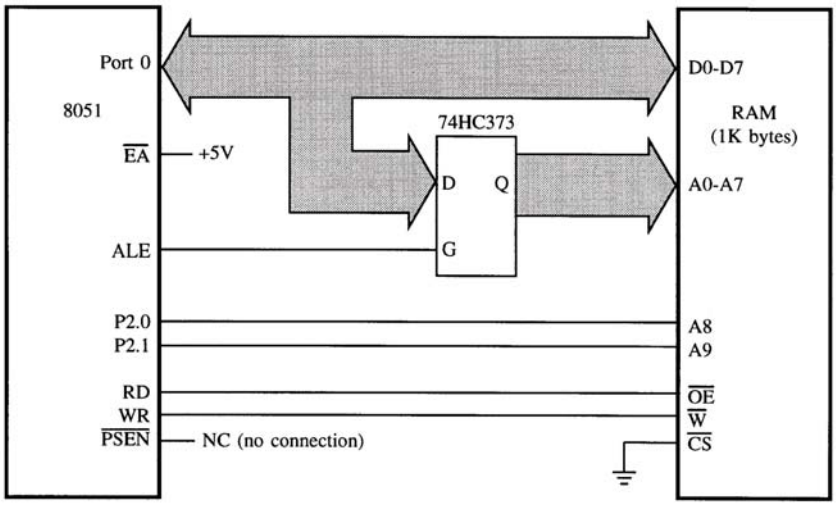
v) Vil CMP-versionen av iBeast-prosessoren være av type "homogeneous" eller "heterogeneous"?

OPPGÅVE 5: DIVERSE BINÆR-QUIZ(10%)

Vel rett svar, sant eller usant. Korrekt svar gir 2 %, feil -1 %, blank eller fleire svar på ei oppgåve gir 0 %.

(i) Ein EPROM kan slettast og reprogramerast. Sant eller usant?

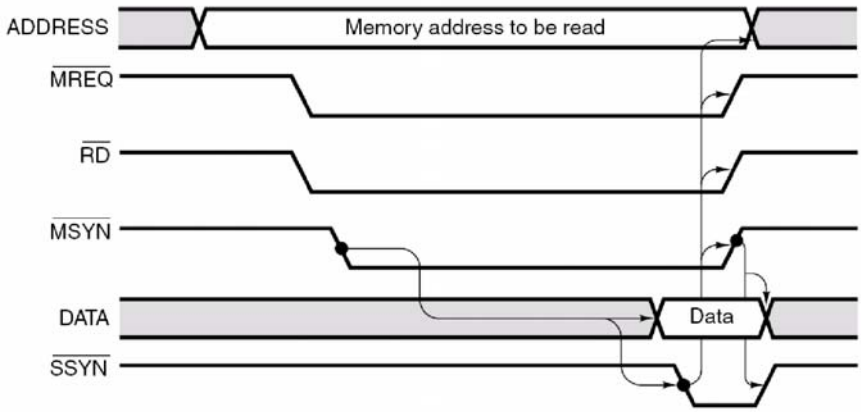
(ii) Figur **??** viser eit system med multiplexa adressebuss og databuss. Sant eller usant?



Figur 5: Bus interface.

(iii) SRAM er raskare enn DRAM. Sant eller usant?

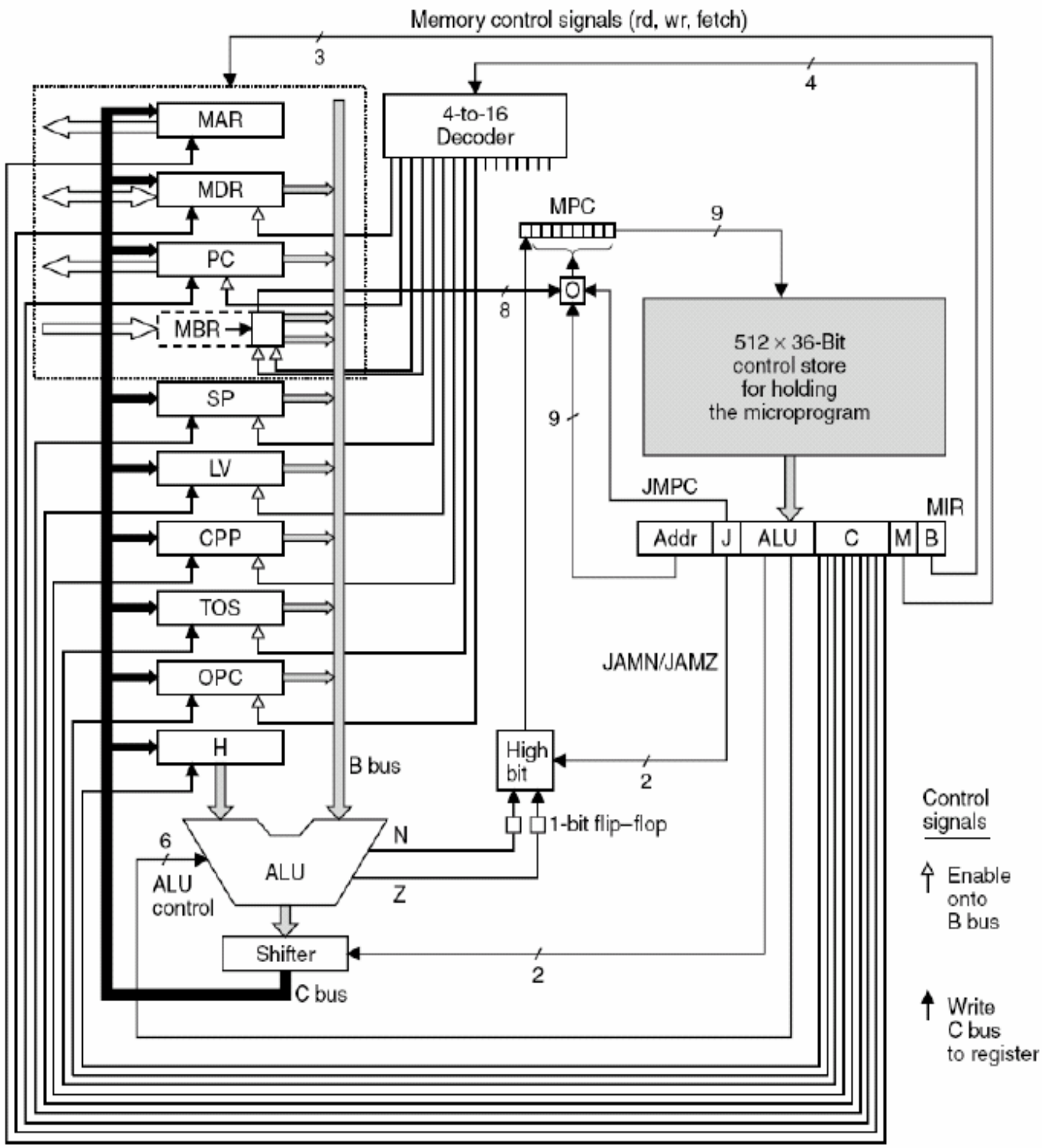
(iv) Figur **??** er eit døme på synkron bussoverføring. Sant eller usant?



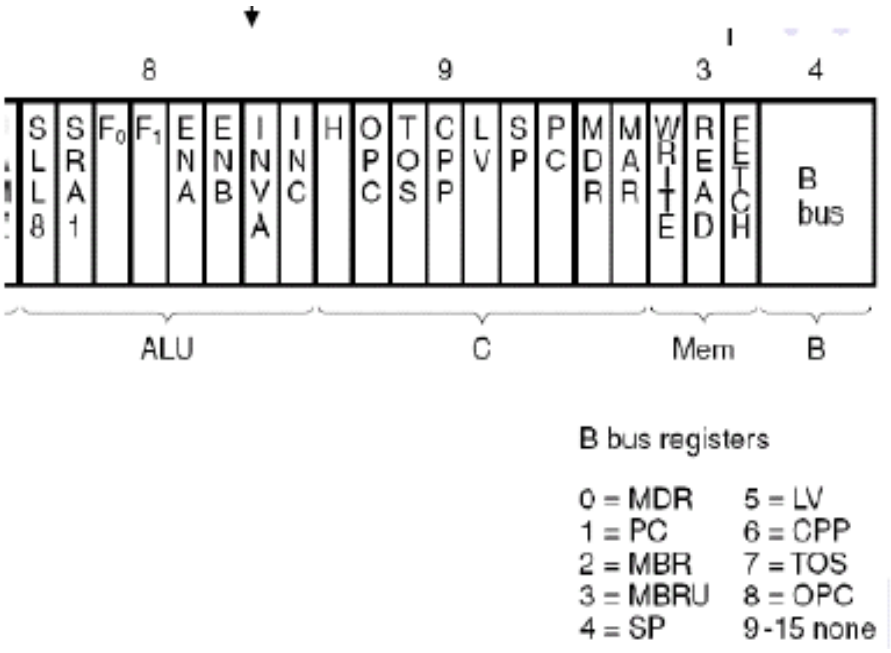
Figur 6: Bus transfer.

(v) Innføring av hurtigbuffer (cache) i minnehirarkiet er ein måte å auke den totale mengda eksternt minne som kan adresserast. Sant eller usant?

IJVM appendix



Figur 7: Block diagram (IJVM).



Figur 8: Microinstruction format (IJVM).

Answer Key for The Exam

OPPGÅVE 1: DIGITAL LOGISK NIVÅ (25% (5% ON A, 10% ON B AND C))

I figur **??** er det eksterne bussgrensesnittet for ein mikrokontroller vist. Det er brukt to RAM-brikker på 32kB og ein ROM-brikke på 1kB. Adressedekodingen er bygd opp av OR-portar og inverterportar. Alle einingane har eit aktivt lågt (logisk ”0”) CS (Chip Select)-signal.

(i) Kva er det maksimale eksterne minneområdet mikrokontrolleren kan adressere?

Answer: 64kB (2^{16})

(ii) Kva er adresseområdet for RAM 1-, RAM 2- og ROM-brikken? Teikn minnekart for systemet.

Answer: ROM 0000 - 00FF
RAM1 0100 - 7FFF
RAM2 8000 - FFFF

(iii)

(iv) For å få ned produksjonskostnaden for systemet, er det foreslått eit nytt maskinvaredesign, sjå figur **??**. I det nye designet er dei to RAM-brikkane erstatta av ein RAM-brikke som er dobbelt så stor.

i) Er minnekartet for systema i figur **??** og figur **??** identiske?

ii) Er det mogleg å behalde programvare utvikla for systemet i figur **??** utan endringar på det nye maskinvaredesignet? Forklar.

Answer: i) Ja dei to minnekarta er identiske.

ii) Ja det er mulig mikrokontrolleren har dei samme minneadressene tilgjengeleg. Det einaste som er forskjelig er at dekodinga er for ein brikke. Det gjer ingen forskjell for programm som er utvikla for systemet i fig. 1. sidan systemet sin minnemodell er lik.

OPPGÅVE 2: MIKROARKITEKTUR OG MIKROINSTRUKSJONAR (15%)

Bruk vedlagte diagram i figur **??**, figur **??**, figur **??** og figur **??** for IJVM til å løyse oppgåvene.

i. Komponenten 4-to-16 Decoder i figur **??** er brukt til å kontrollere kva som skal liggje på B-bus. For C-Busen er separate bit brukt til å kontrollere kvart register. Kvifor?

Answer: Ikkje mulig å legge ut fleire register på felles buss (B-Buss), vil skape konflikt, derfor er kontroll av B-Buss koda med 4 bit i MIR, sparar også plass i control store. C-buss verdien er mulig å lagre i fleire register, derfor seperate bit for kontroll av kvart register.

Lag mikroinstruksjon(ar) for følgjande IJVM-operasjon: *OPC = TOS - LV + 1*.

Sjå vekk frå Addr- og J-felta i mikroinstruksjonsformatet. Angi korrekte bit for ALU, C, Mem og B gitt i figur **??**.

ii. **Answer:** Her er det mange løysingar, alt etter kva valg ein gjer for kvar microinstruksjon, viktigt at ein ser at det må være fleire sidan ein må bruke H-register til mellom lagring og at ein stokkar rett når ein brukar B - A ALU-funksjon
Eksempel løysing 1:
LV -> H: ALU: 010100 (B) C: 100000000 (H) Mem: 000 B: 0101 (5)
TOS - H -> H: ALU: 111111 (B - A) C: 100000000 (H) Mem: 000 B: 0111 (7)
H + 1 -> OPC: ALU: 010100 (A + 1) C: 010000000 (OPC) Mem: 000 B: 1001 (none)
Har då gjort OPC = (TOS - LV) + 1 med 3 mikroinstruksjonar.
Eksempel løysing 2:
LV + 1 -> H: ALU: 110111 (B + 1) C: 100000000 (H) Mem: 000 B: 0101 (5)
TOS - H -> OPC: ALU: 111111 (B - A) C: 010000000 (OPC) Mem: 000 B: 0111 (7)
Har då gjort OPC = TOS - (LV + 1) med 2 mikroinstruksjonar.
Det er fleire løysingar, men som vist i eksempel må TOS og LV mellomlagrast rett for å nytte B - A ALU-funksjon.

OPPGÅVE 3: INSTRUKSJONSSETT ARKITEKTUR(ISA)(25%)

Ein særs enkel prosessor har ein ”load”, ein ”store”, 8 ALU-instruksjonar og nokre spesialinstruksjonar, inkludert NOP-instruksjonen. Instruksjonsformatet for instruksjonane er vist i figur **??**. Alle register og bussar er 32-bit. Prosessoren er av typen "Harvard architecture".

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, $Rn = Rn + Rn$.
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, $Rn = Rn - 1$.
MUL Rn, Rn, Rn Multiplication, $Rn = Rn * Rn$.
CMP, Rn, Rn Compare, Z-flag = 1 if $Rn = Rn$, else Z-flag = 0.

Special: Misc.

CP Rn, Rn Copy, $Rn < -Rn$.
NOP Waste of time, 1 clk cycle.
MOVC Rn, constant Put a constant in register, $Rn = C$.
LDM Rn, Adr Load Rn with content of memory Address.
SDM Rn, Adr Store content of Rn in memory Address.

Rn: Any user register.
DC: Don't care.
C: Constant.
Adr: 16 bit address space, 0000 0000 - 0000 FFFF.

i. Basert på tilgjengeleg informasjon kan ein ikkje definera denne prosessoren til å være generell.

   i) Kvifor?

   ii) Kva må til for at prosessoren skal være generell? Gi eit døme.

Answer: i) Manglar instruksjon for conditional branch (flow control, betinga hopp).

ii) Legg til conditional Branch. Har oppgitt at det er eit Z-flagg tilgjengeleg (CMP-instruksjon). Kan då bruke CMP til å utføre test. Legg då til ein branch instruksjon, f.eks. branch on zero; BZ Rn, hopp til adr gitt i Rn viss Z = 1. (PC = Rn), viss Z = 0 utfør neste instruksjon. (Her er det mange mulige eksempel løysingar, men må få til betinga hopp.

ii. NOP-instruksjonen er ei 0-adresseinstruksjon (Zero-address instruction). Kva andre adresseringar (addressing) er brukt i instruksjonssettet?

Answer: Denne møten å klasifisere instruksjonar på (Zero-address instruction) angir operandar som er i bruk, 0 til n. For denne maskina er det i tillegg til Zero-address:
Two-address: e.g. CMP
Three-address: e.g. Add

iii. MOVC-instruksjonen flytter ein konstant verdi inn i eit register.

   i) Kva adresseringsmodus (addressing mode) er brukt i MOVC-instruksjonen?

   ii) Når kan verdien til konstanten definerast?

Answer: i) Immidiate Addressing modi. konstanten er ein immidiate operand. Operanden blir henta frå programminne som ein del av instruksjonen.

ii) Immidiate operandar må då definerast i koden (Compile time) altså når programme blir laga. (Sidan prosessoren er ein Harvard prosessor er det heller ikkje mulig å endre verdien på konstanten med ”self-modifying code").

iv. Er det mogleg å bestemme om denne prosessoren nyttar samleband (pipelining) utfrå tilgjengeleg informasjon? Forklar kvifor / kvifor ikkje.

Answer: Nei, ingen detaljar om korleis implemantasjonen er gjort prosessoren er skissert på ISA-nivå.

v. Kva er det totale minneområdet prosessoren kan adressera?

Answer: 2^{32}. dette er område som Load og Store instruksjonane kan adressere. 2^{32} sidan alle register og bussar er 32 bit (LDM og SDM er instruksjonar nyttar eit Direkte Addresing modi, men kan kunn nå 2^{16} lokasjonar frå 0x0000 0000 - 0x0000 FFFF. Så desse instruksjonane kan kunn nå eit sub-sett av det totale minneområde (det er ein F formykje i oppgåvesette for oppgit *Adr* felt, ingen innvirking på total adr. minneområde).

OPPGÅVE 4: DATAMASKINER OG YTING(25% (10% ON A; 15% ON B))

i. Figure **??** and Figure **??** i appendix viser ulike versjonar av IJVM-mikroarkitekturen. Den orginale mikroarkitekturen er vist i figur **??**.

   i) Vil endringen i mikroarkitektur frå figur **??** til figur **??** kreve endringar i MIR? Forklar.

   ii) I figur **??** er samleband innført. Er superscalarity ein eigenskap som er tilstades i mikroarkitekturen vist i figur **??**?

   iii) Mikroarkitekturane vist i figur **??** og figur **??** krever også endringar i mikroinstruksjonane. Vil ISA-nivået bli påverka av at ein endrar korleis instruksjonar vert utført på mikroarkitekturnivå?

   Answer: i) Må no også ha kontroll signal for å legje register innhald ut på A-Buss (kan då gjere som for B-Buss 4 bit og 4-to-16 dekoder).

   ii) Har lagt til pipelining, men dupliserar ingen utførande einingar, ingen sann parallellitet. Så MIC 3 er ikkje ein superscalar prosessor.

   iii) Nei, ISA vil ikkje bli påvirka, instruksjonsett og minnemodell er heilt lik.

ii. I ein tenkt prosessorfamilie, iBeast, eksisterar det to versjonar: iAardvark og iArmadillo. iAardvark er eit trestegs samlebanddesign. iArmadillo er eit femstegs samlebanddesign. Figur **??** viser dei to samlebanda med trinnforsinking.

   i) Estimer maksimal klokkefrekvens for iArmadillo og iAardvark.

   ii) Estimer klokkefrekvens for ein iBeast-prosessorversjon utan samleband.

   iii) Prosessorfamilien er produsert i ein 22nm-prosess. Korleis vil mest truleg ein nedskalering til ein 18nm-produksjonsprosess påverke trinnforsinkelsen?

   iv) Lat oss si at du vil lage ein Chip Multi Processor (CMP)-version av iBeast-prosessoren. Kva prosessorkjerne, iArmadillo eller iAardvark, vil være det beste valget viss høg ILP er målet? Forklar.

   v) Vil CMP-versionen av iBeast-prosessoren være av type "homogeneous" eller "heterogeneous"?

   Answer: i) $f = 1/t$ (t lengste trinn). $f = 1/20ns$ for iAardvark og $f = 1/7.5ns$ for iArmadillo.

   ii) Viss alle samlebandstega fjernast må alle trinn fortsatt utførast. Får då eit estimat på $f = 1/30ns$

   iii) Trinnforsinkelsen vil gå ned. (dette er henta rett ut frå kvifor Moores lov virkar på ytelse (hastigheit) også. Når ein skalerar ned vil transistorar kunne toggle raskare (mindre kapasitans)).

iv) iAardvark proseserar 3 inst. samstundes og iArmadillo proseserar 5 inst samstundes (5 trinn (5 mot 3)) så iArmadillo er best viss høg Instruction Level Paralellism (ILP) er målet.

v) Homogeneous (kunn like kjerner).

OPPGÅVE 5: DIVERSE BINÆR-QUIZ(10%)

Vel rett svar, sant eller usant. Korrekt svar gir 2 %, feil -1 %, blank eller fleire svar på ei oppgåve gir 0 %.

A. Ein EPROM kan slettast og reprogramerast. Sant eller usant?

   Answer: Sant.

B. Figur **??** viser eit system med multiplexa adressebuss og databuss. Sant eller usant?

   Answer: Sant.

C. SRAM er raskare enn DRAM. Sant eller usant?

   Answer: Sant.

D. Figur **??** er eit døme på synkron bussoverføring. Sant eller usant?

   Answer: Usant, det er ingen klokke.

E. Innføring av hurtigbuffer (cache) i minnehirarkiet er ein måte å auke den totale mengda eksternt minne som kan adresserast. Sant eller usant?

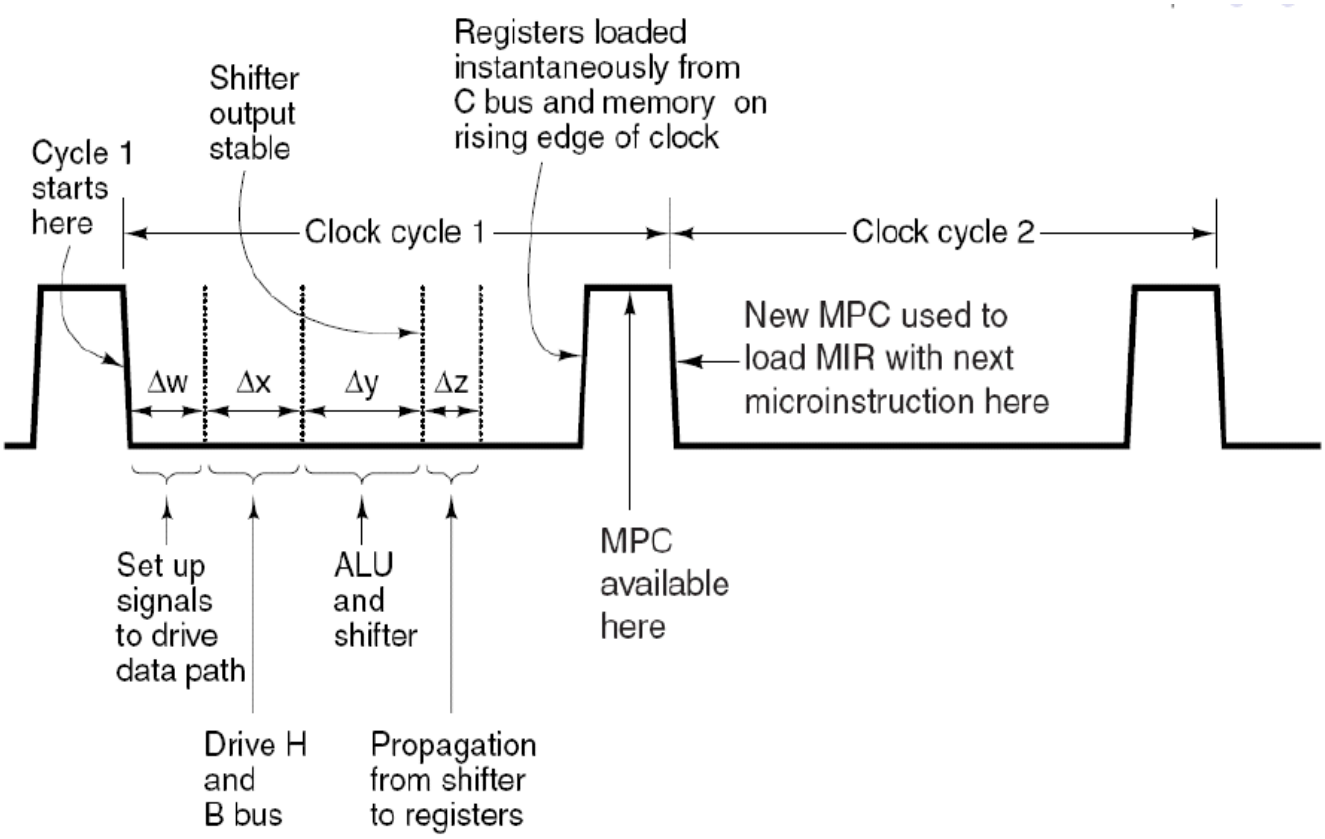
   Answer: Usant (cache er kunn ein avbilding (buffer)).

IJVM appendix

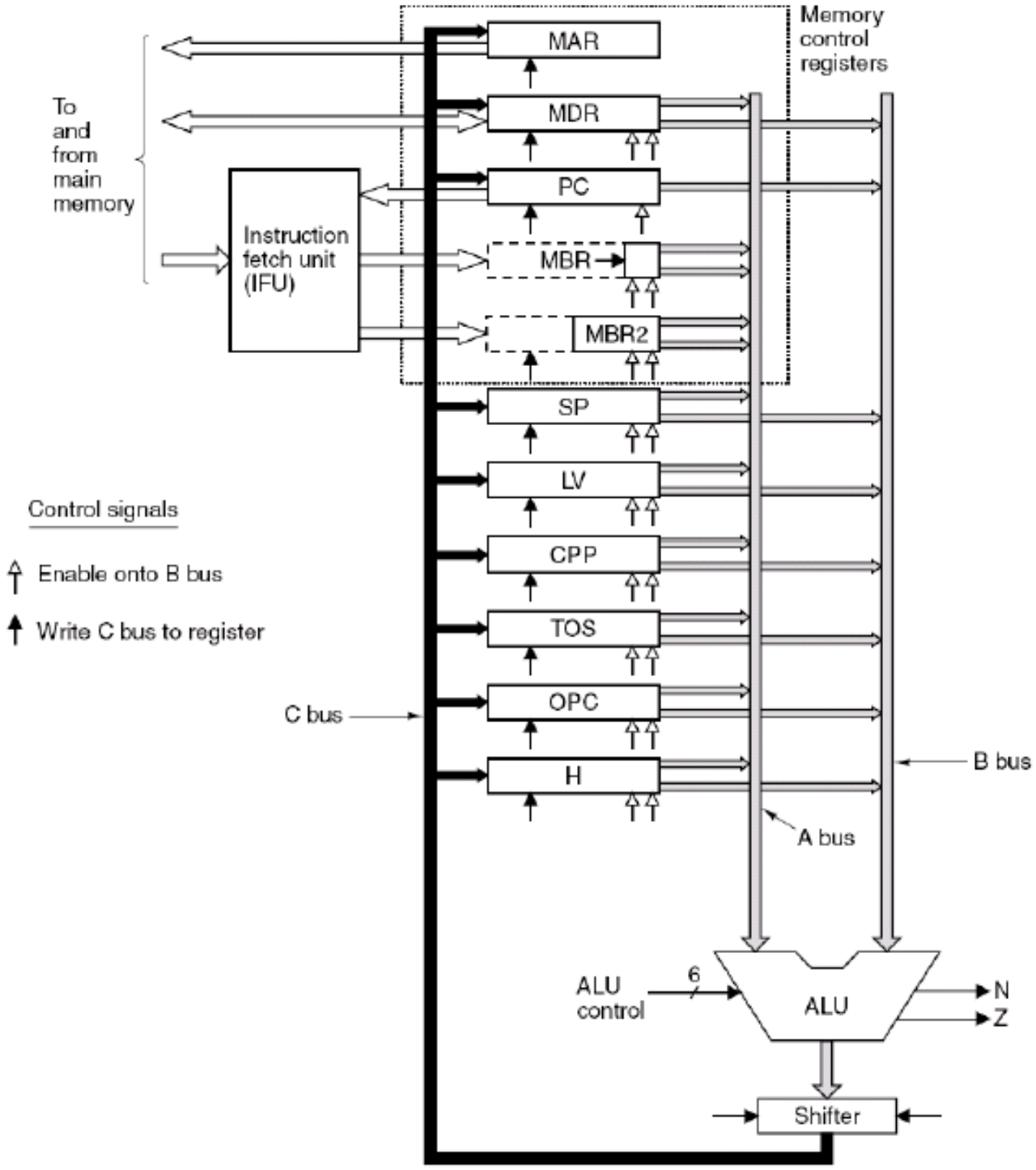
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

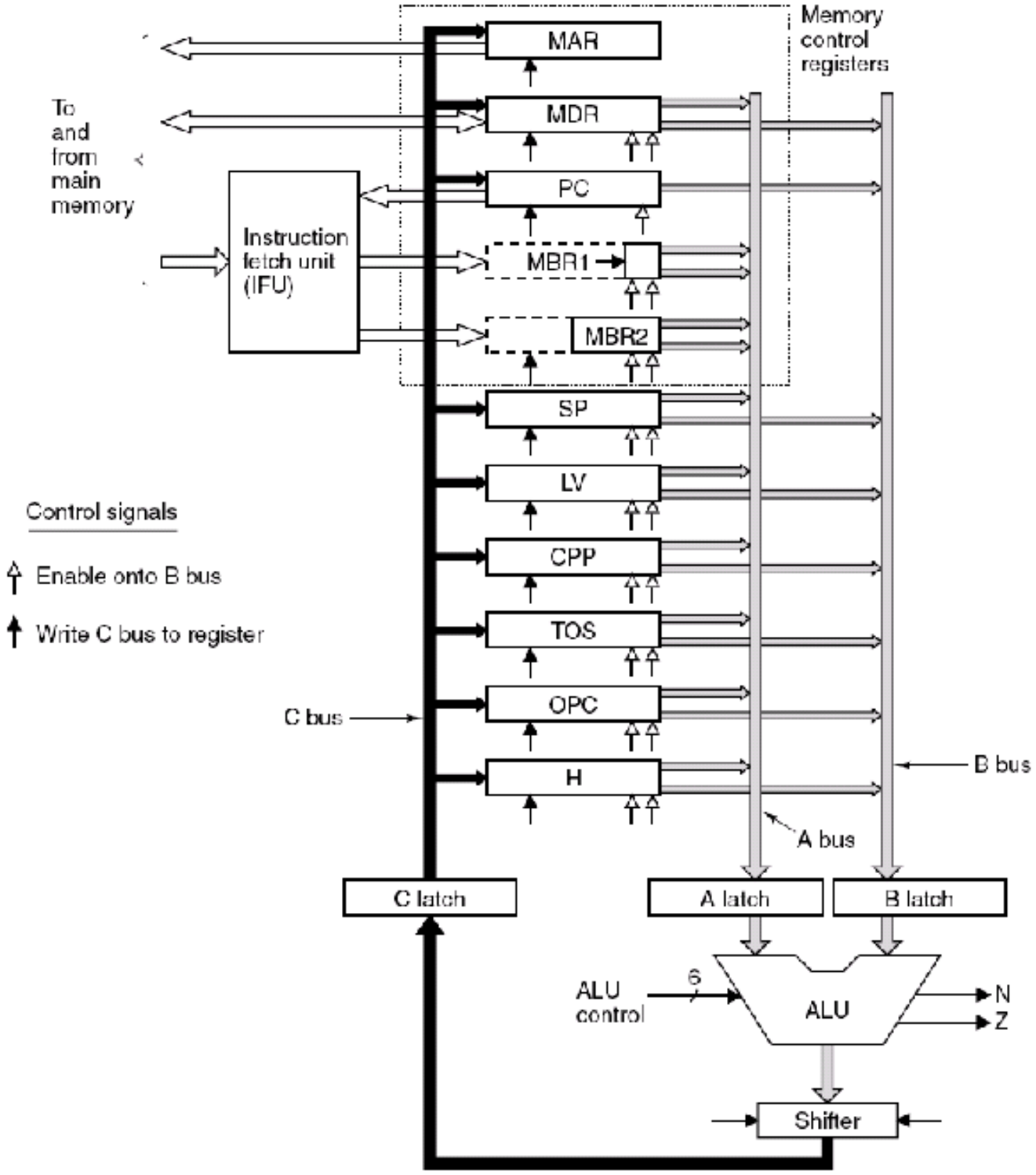
Figur 9: ALU functions (IJVM).



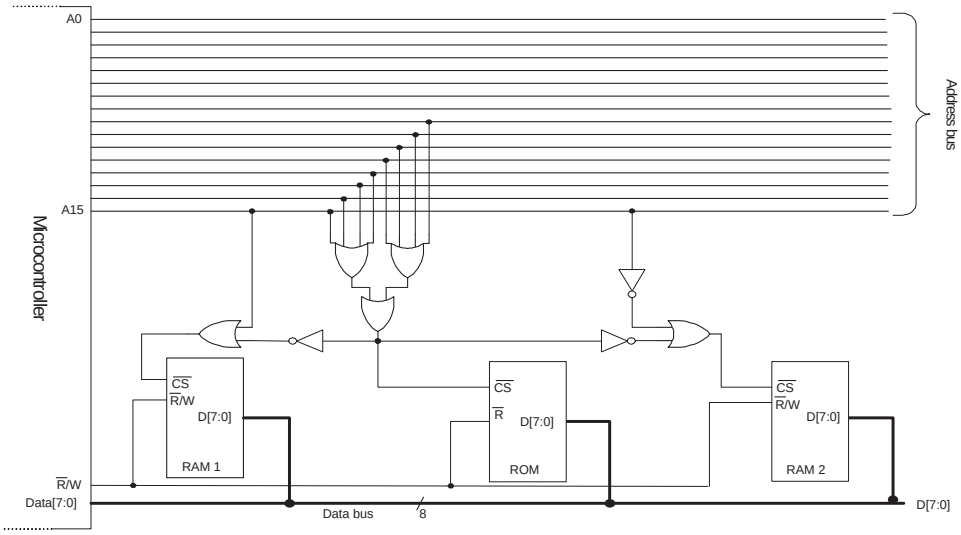
Figur 10: Timing diagram (IJVM).



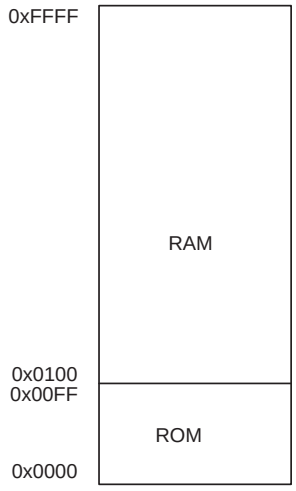
Figur 11: Alternative microarchitecture I.



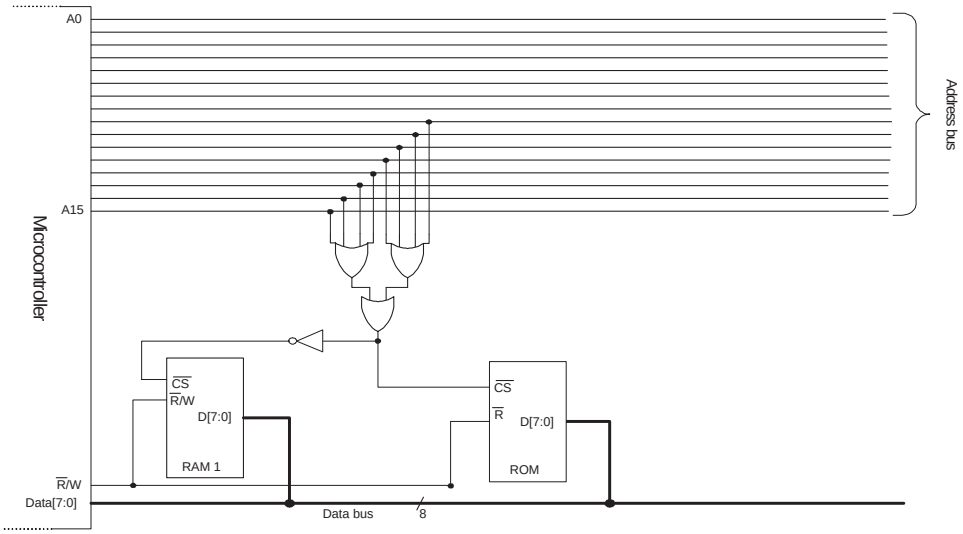
Figur 12: Alternative microarchitecture II.



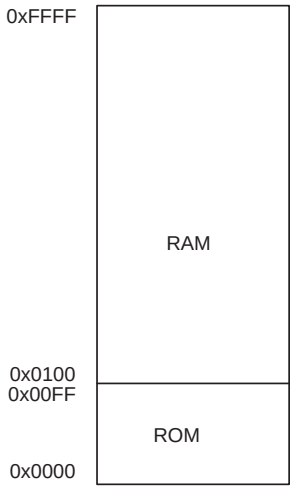
Figur 13: Address decoding.



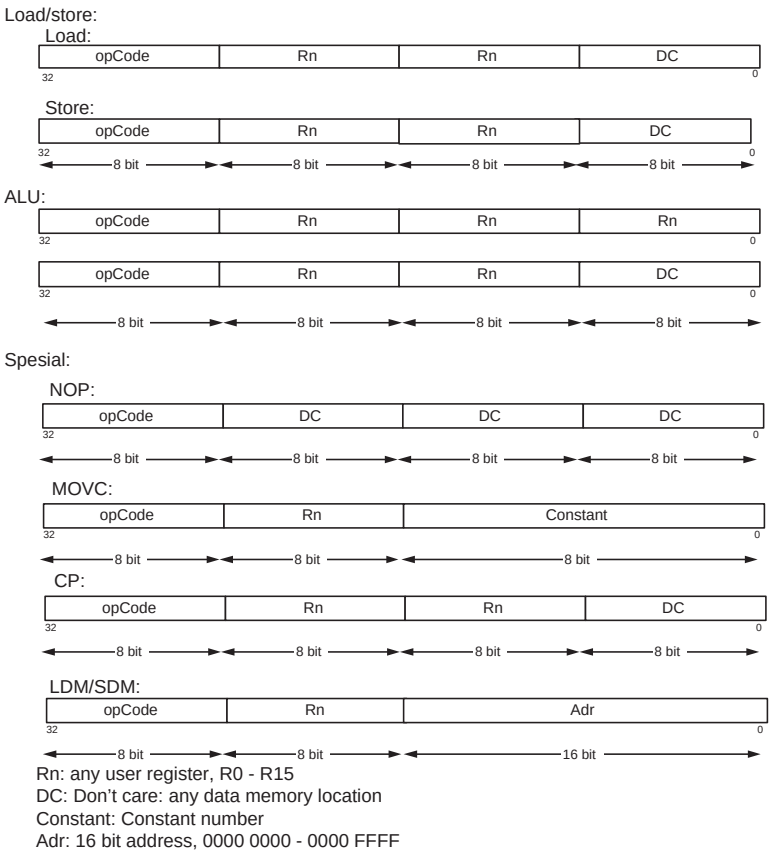
Figur 14: Memmory map for fig. 1.



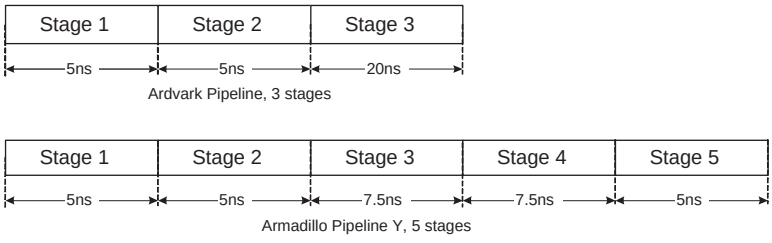
Figur 15: New address decoding.



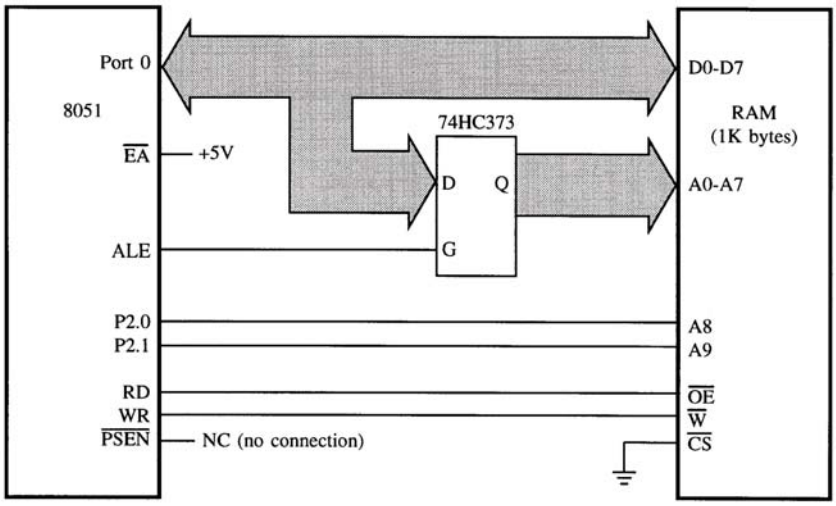
Figur 16: Memmory map for fig. 2.



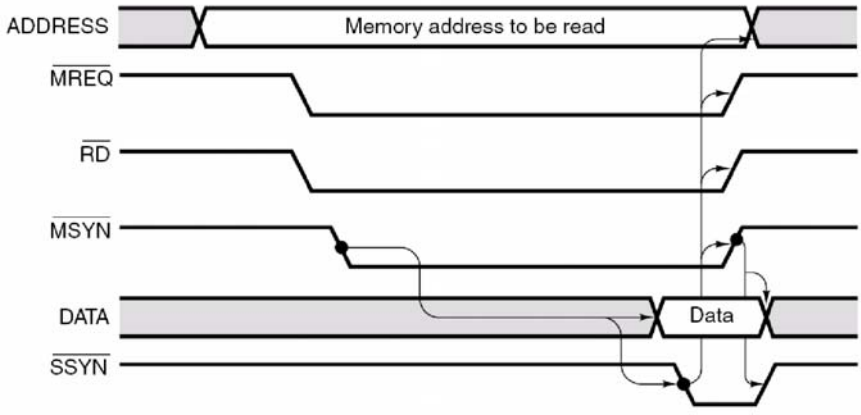
Figur 17: Instruksjonsformat.



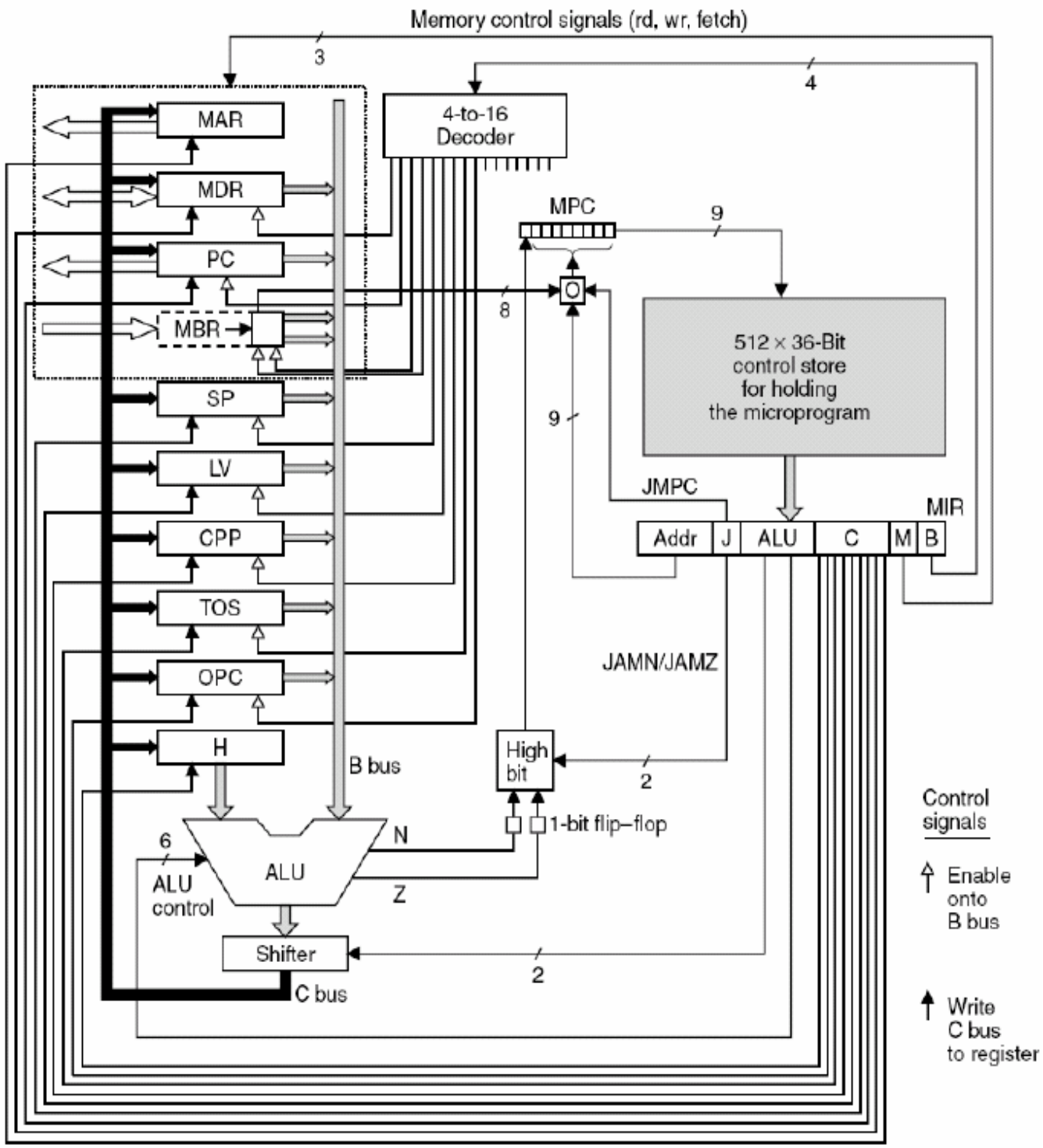
Figur 18: Samlebandtidsbruk for iArmadillo- og iAardvark-versionane av iBeast.



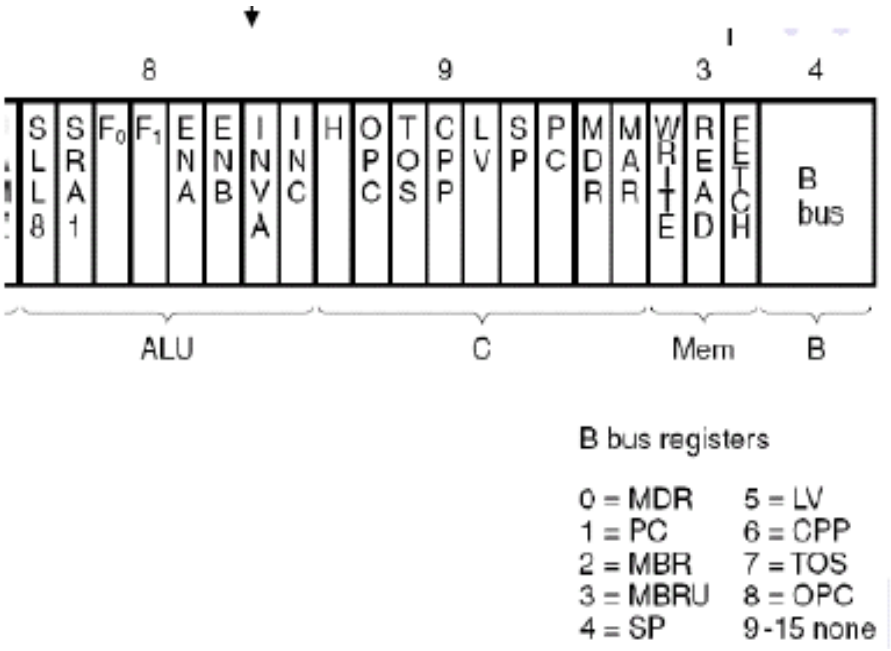
Figur 19: Bus interface.



Figur 20: Bus transfer.



Figur 21: Block diagram (IJVM).

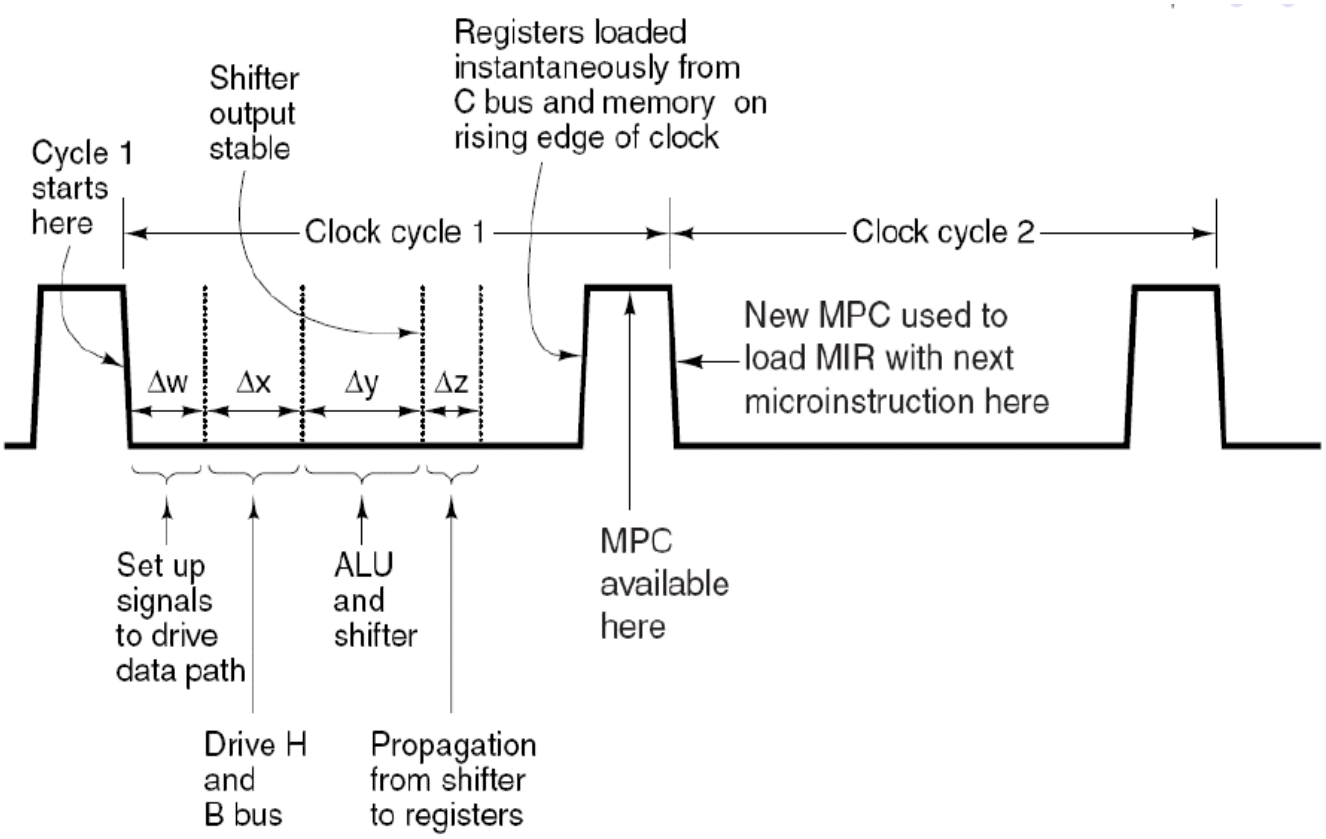


Figur 22: Microinstruction format (IJVM).

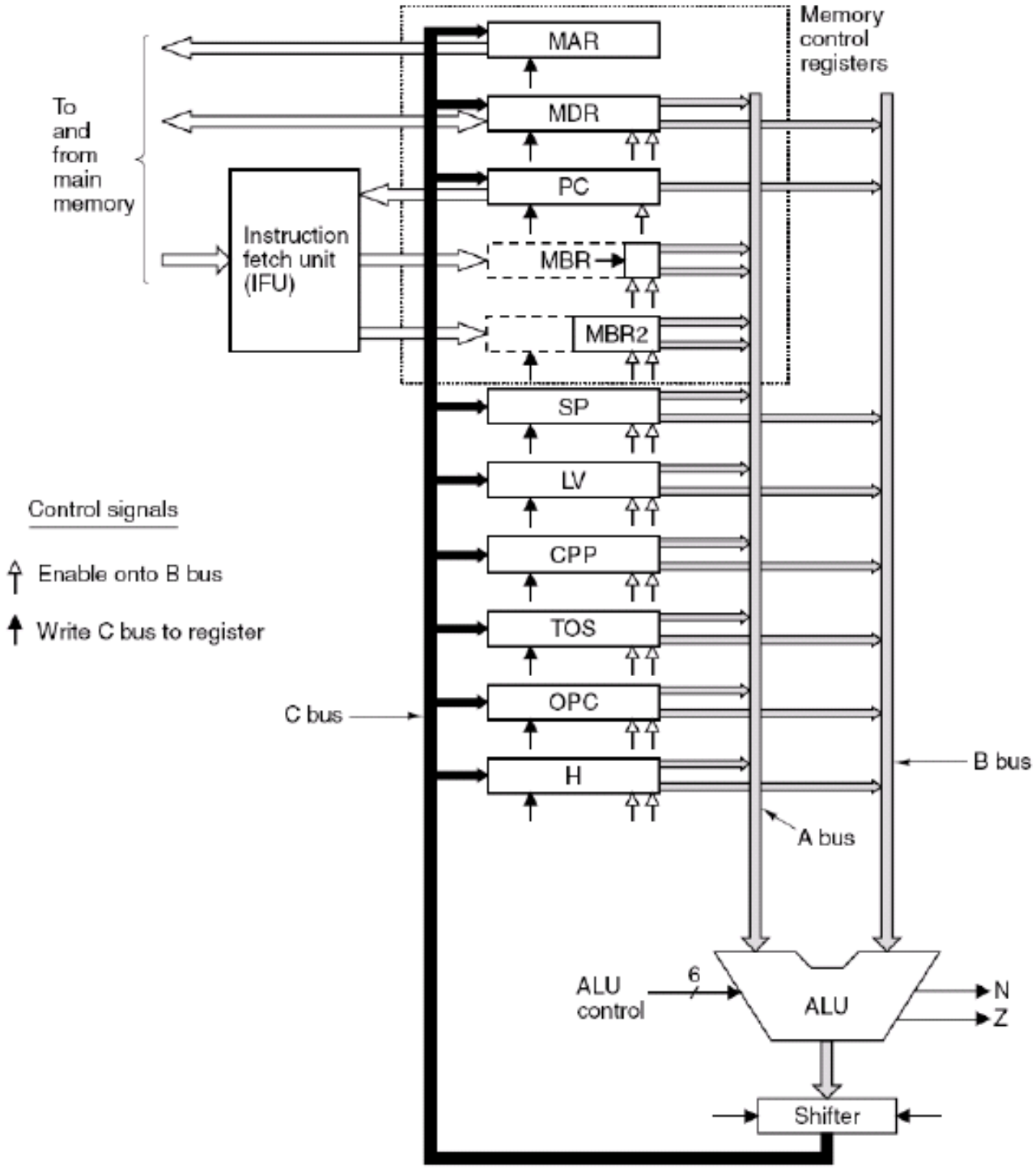
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

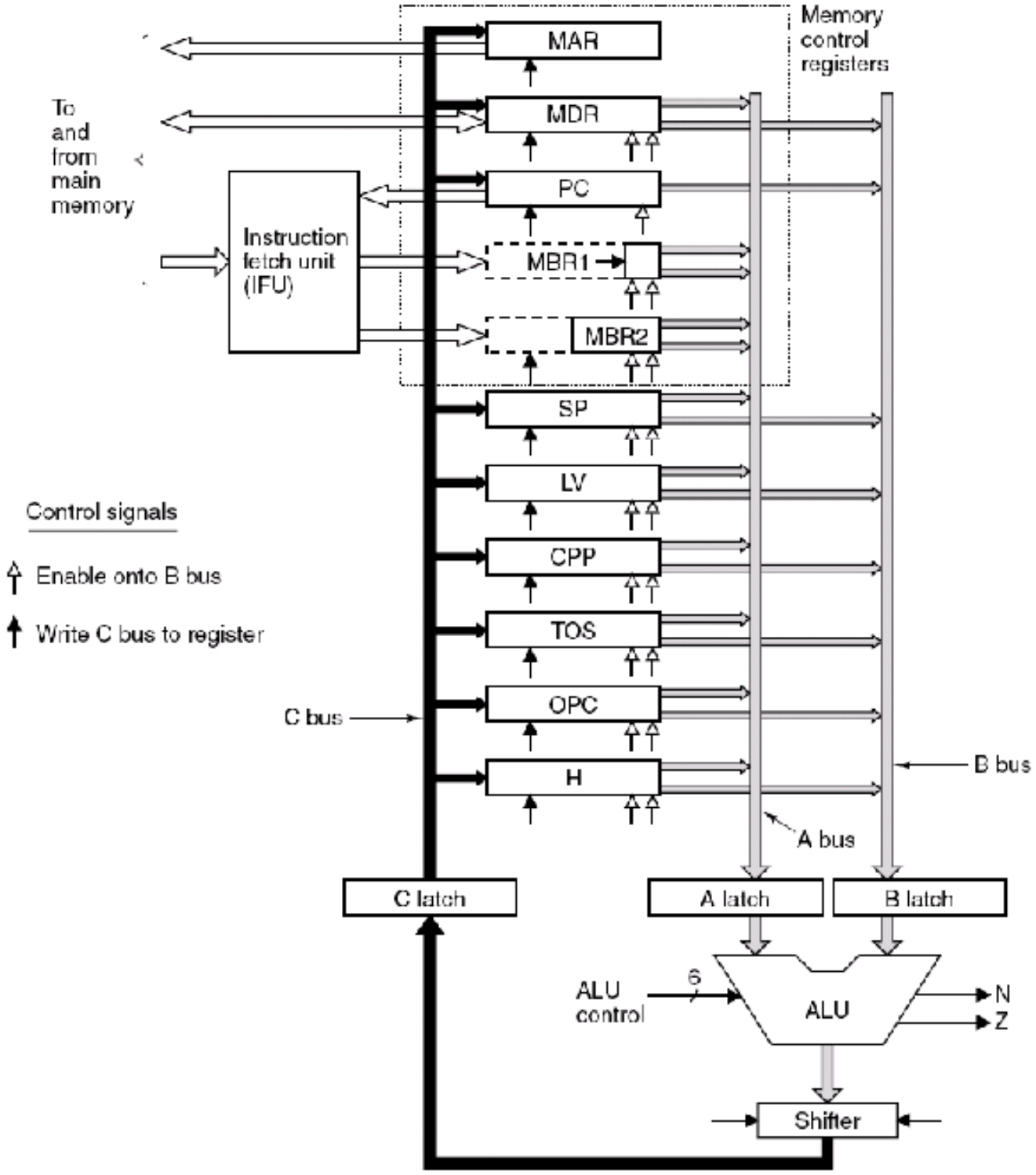
Figur 23: ALU functions (IJVM).



Figur 24: Timing diagram (IJVM).



Figur 25: Alternative microarchitecture I.



Figur 26: Alternative microarchitecture II.