



TDT4160

DATAMASKINER GRUNNKURS

EKSAMEN

17. DESEMBER, 2012, 09:00–13:00

Kontakt under eksamen:

Gunnar Tufte 73590356 / 97402478

Tillatte hjelpemidler:

D.

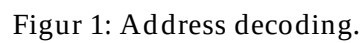
Ingen trykte eller håndskrivne hjelpemiddel er tillete.

Enkel godkjent kalkulator er tillete.

Målform:

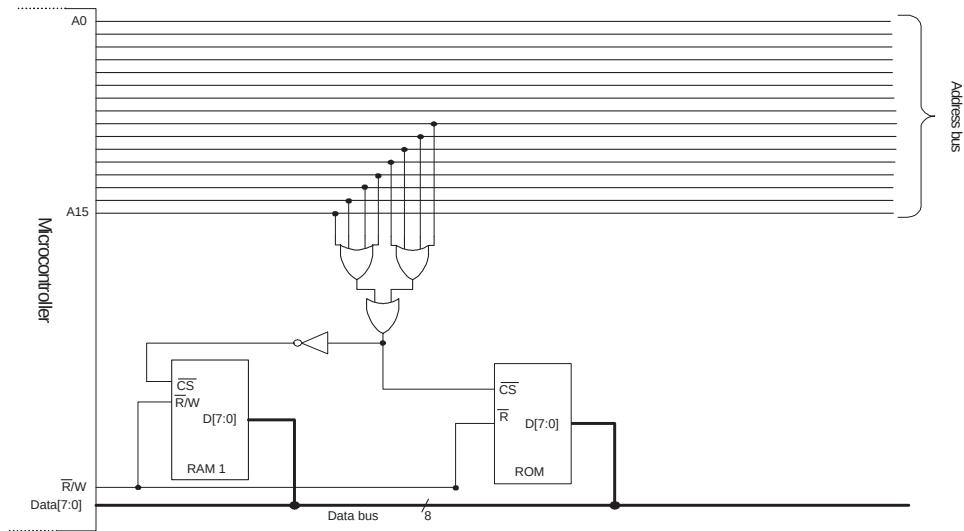
Nynorsk

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.



- Page 2 of ??

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.
- Er minnekartet for systema i figur ?? og figur ??? identiske?
 - 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?

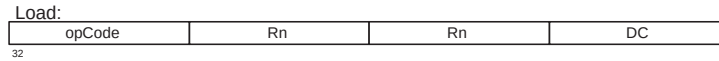
(ii) Lag mikroinstruksjon(ar) for følgjande IJVM-operasjon: $OPC = TOS - LV + 1$.

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

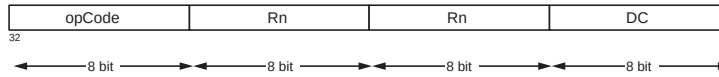
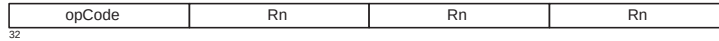
OPPGÅVE 3: INSTRUKSJONSSETT ARKITEKTUR(ISA)(25%)

Ein sær enkel prosessor har ein "load", ein "store", 8 ALU-instruksjonar og nokre spesial-instruksjonar, inkludert NOP-instruksjonen. Instruksjonsformatet for instruksjonane er vist i figur ??.

Load/store:



ALU:



Spesial:



Rn: any user register, R0 - R15

DC: Don't care: any data memory location

Constant: Constant number

Adr: 16 bit address, 0000 0000 - 0000 FFFF

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 Bitwise OR, $Rn = Rn + Rn$.

INV Rn, Rn Bitwise 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.

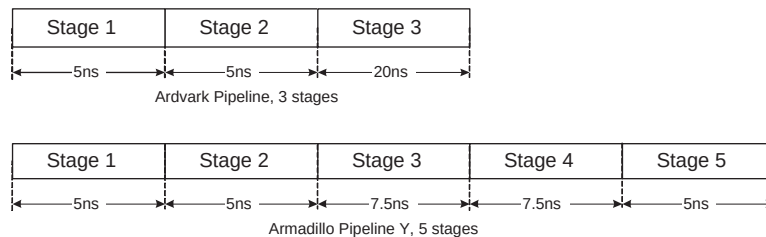
Adr: 16 bit address space, 0000 0000 - 0000 FFFF.

- (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 originale 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.



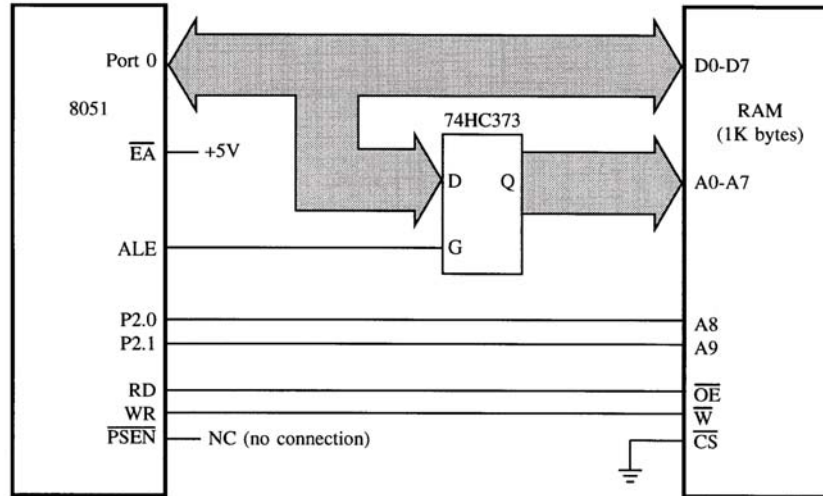
Figur 4: Samlebandtidsbruk for iArmadillo- og iAardvark-versjonane av iBeast.

- 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-versjonen av iBeast-prosessoren være av type "homogeneous" eller "heterogeneous"?

OPPGÅVE 5: DIVERSE BINÆR-QUIZ(10%)

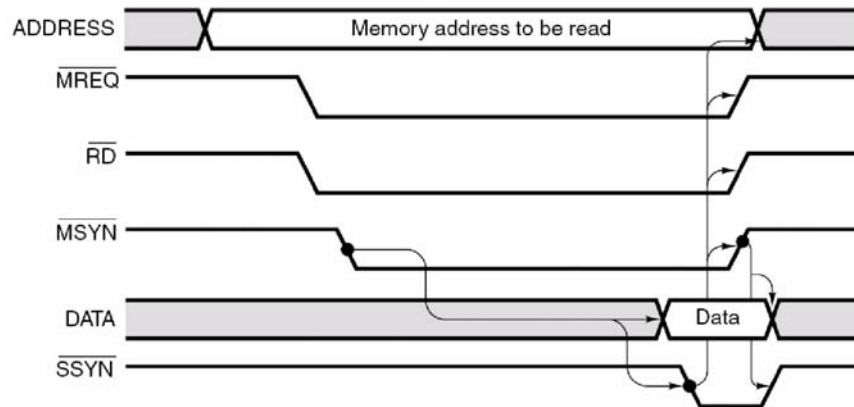
Vel rett svar, sant eller usant. Korrekt svar gir 2 %, feil -1 %, blank eller fleire svar på ei oppgave gir 0 %.

- (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.

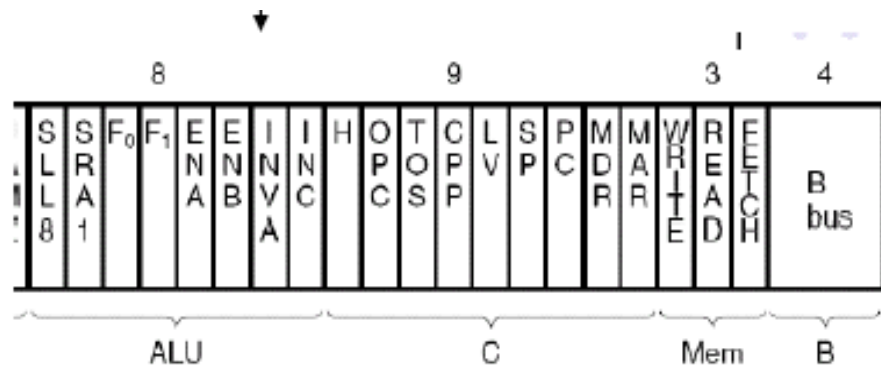
- (iii) SRAM er raskare enn DRAM. Sant eller usant?
- (iv) Figur ?? er eit døme på synkron bussoverføring. Sant eller usant?



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



B bus registers

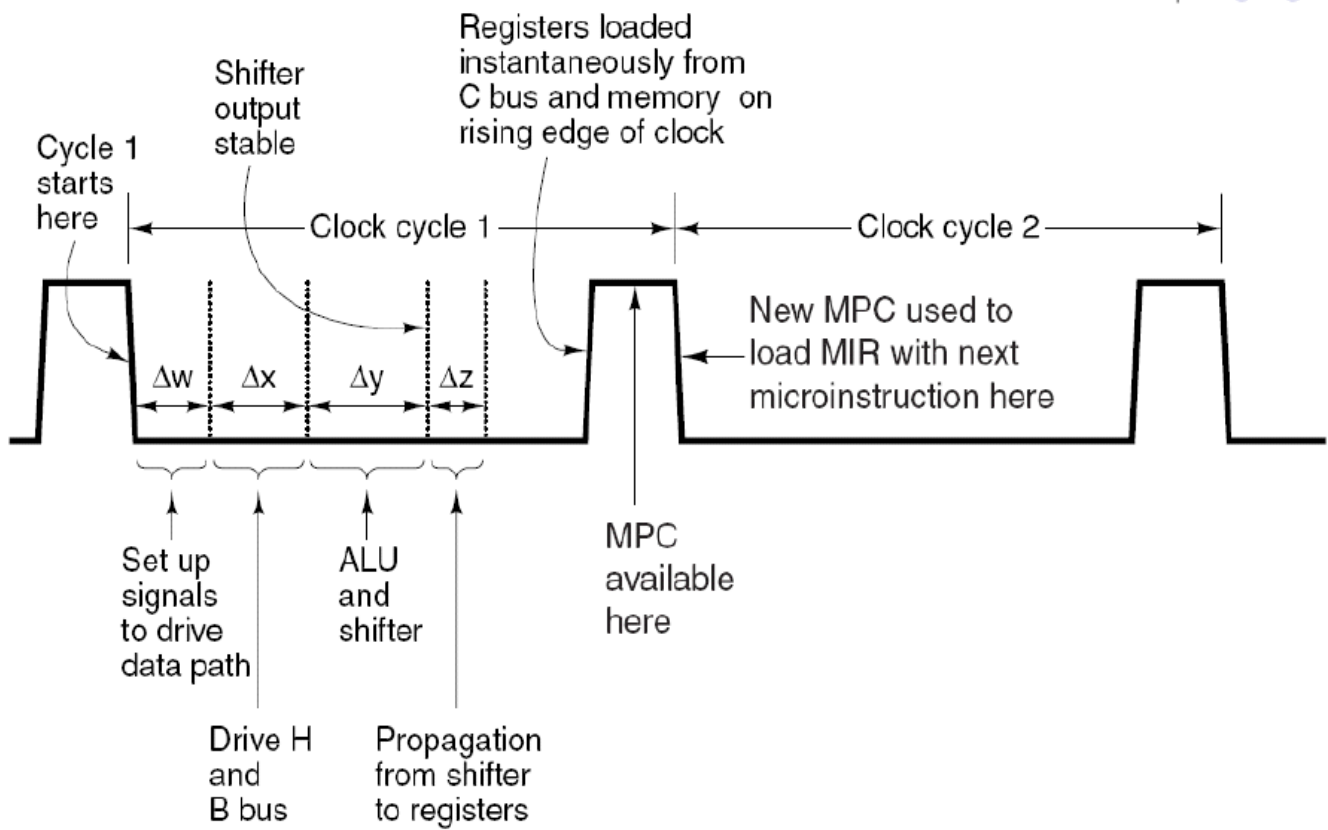
0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Figur 8: Microinstruction format (IJVM).

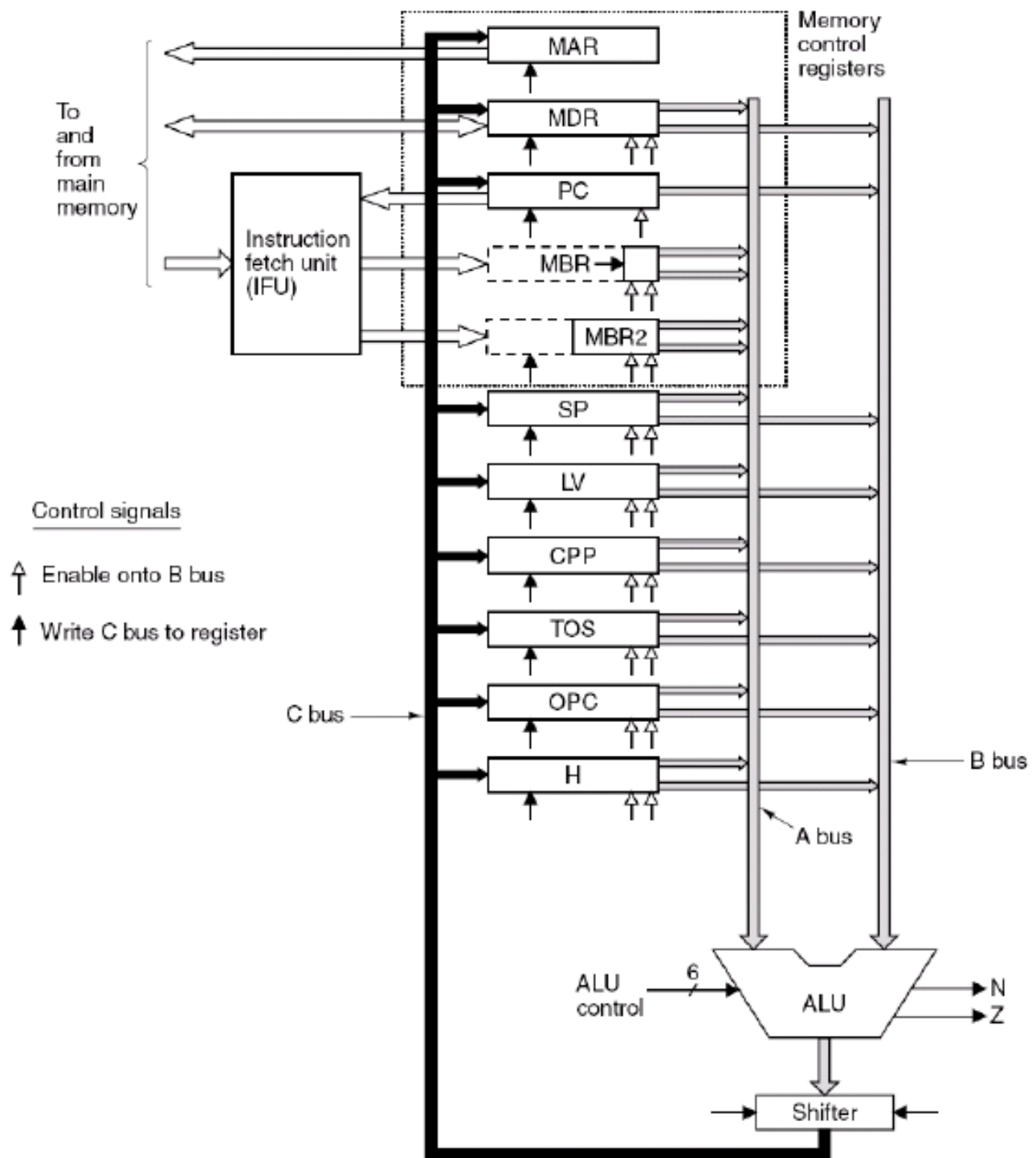
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B - A
1	1	0	1	1	0	B - 1
1	1	1	0	1	1	-A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1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

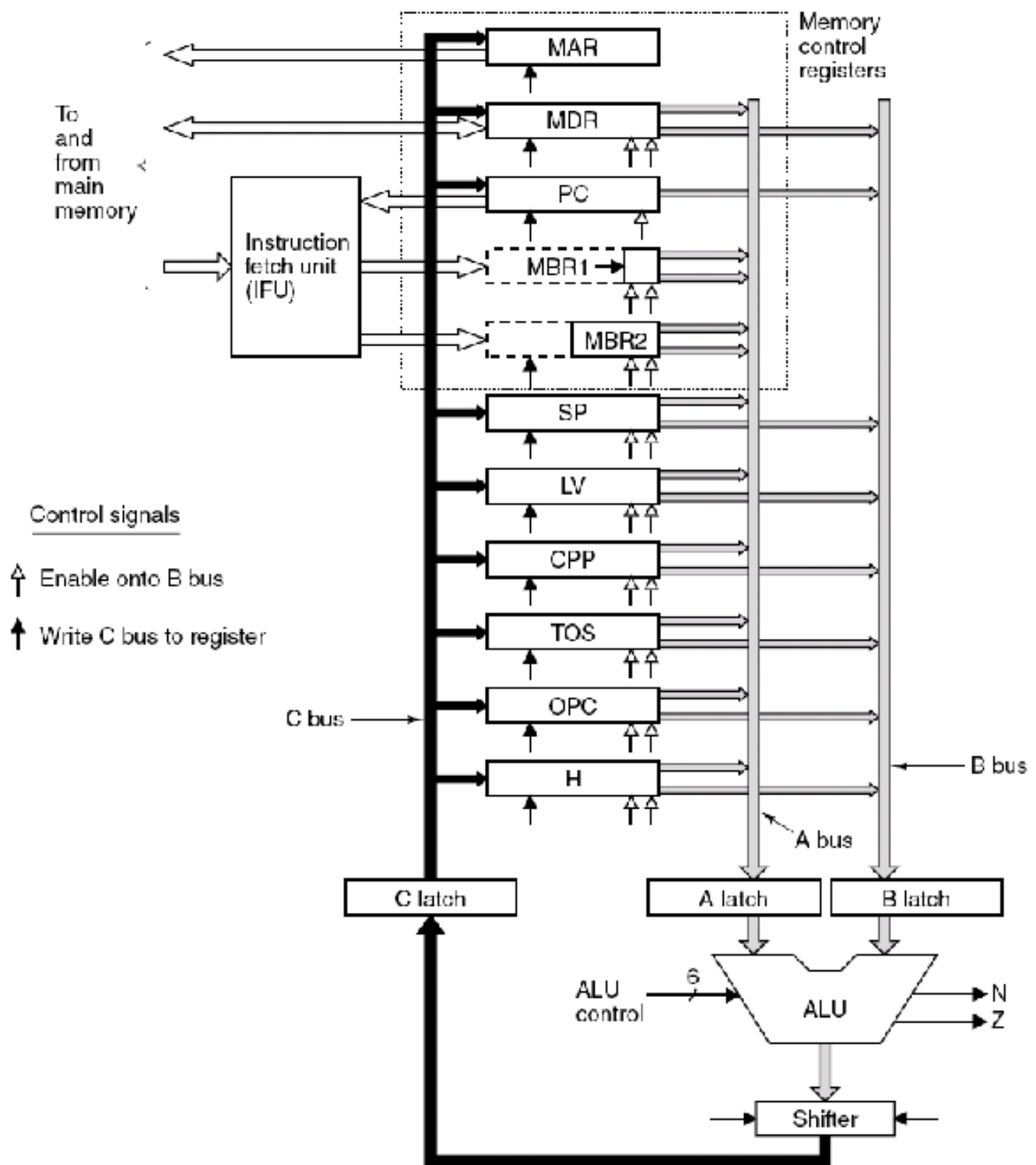
Figur 9: ALU functions (IJVM).



Figur 10: Timing diagram (IJVM).



Figur 11: Alternative microarchitecture I.



Figur 12: Alternative microarchitecture II.