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

10. DESEMBER, 2008, 09:00–13:00

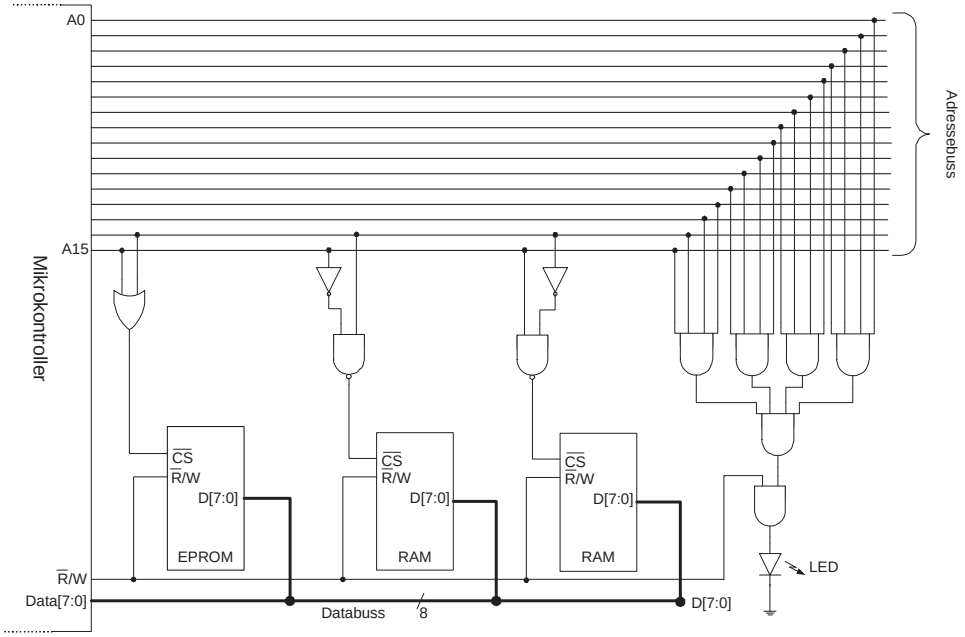
Kontakt under eksamen:
Gunnar Tufte 73590356

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

Målform:
Nynorsk

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

a. I Figur 1 er EPROM, RAM og ein lysdiode (LED) kobla til ein felles buss. Finn adresseområde for einingane. EPROM og RAM har aktivt lågt (logisk ”0”) CS (Chip Select) signal.



Figur 1: Adressedekoding.

b. Er det mogleg å utvide tilgjengeleg RAM med med 16kB (16384 bytes)? Grunngi svaret.

c. Når lyser lysdioden (LED)?

OPPGÅVE 2: MIKROARKITEKTUR OG MIKROINSTRUKSJONAR (20% (5% PÅ A OG B; 10% PÅ C))

Bruk vedlagte diagram i figur 5, figur 6, figur 7 og figur 8 for IJVM til å løyse oppgåvene.

a. Forklar funksjonen til registeret ”MPC”.

b. Lag mikroinstruksjon(ar) for følgjande IJVM-operasjon: last register ”OPC” med innhaldet i register ”H”.

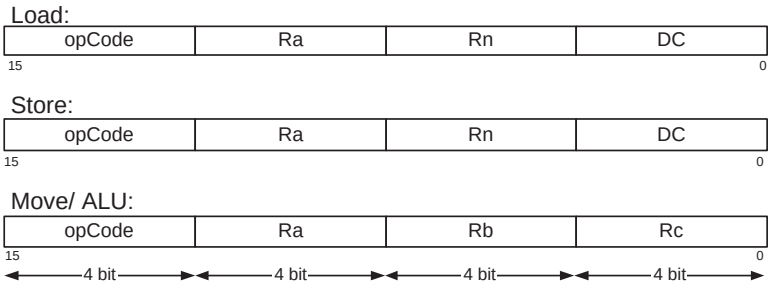
   Sjå vekk frå Addr- og J-felta i mikroinstruksjonsformatet. Angi korrekte bit for ALU, C, Mem og B gitt i Figur 6.

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

   Sjå vekk frå Addr- og J-feltene i mikroinstruksjonsformatet. Angi korrekte bit for ALU, C, Mem og B gitt i Figur 6.

OPPGÅVE 3: INSTRUKSJONSSETT ARKITEKTUR (ISA)(20%)

I ein tenkt 16 bit arkitektur har ein berre tre typar instruksjonar, last (load), lagre (store) og flytt / ALU (Move / ALU). Figur 2 viser dei tre instruksjonstypane.



Load:
- Ra: points to memory location, any user register
- Rn: data destination register, any user register
- DC: Don't Care, not in use

Store:
- Ra: points to memory location, any user register
- Rn: data source register, any user register
- DC: Don't Care, not in use

Move/ ALU::
- Ra: destination register, any user register
- Rb: source register, any user register
- Rc: source register, any user register

Figur 2: Adressedekoding.

a. Utfrå tilgjengeleg informasjon:

   i) Kva er det maksimale antal instruksjonar denne maskina kan ha? grunngi svaret.

   ii) Kva er det maksimale antal brukar register (user registers) denne maskina kan ha?, grunngi svaret.

b. Kva type arkitektur er dette instruksjonsformatet for?

c. Viss ein endrar formatet for load / store til å bruke instruksjonsfeltet merka DC til å innehalde ein index og brukar *Ra* som base adresse har ein endra addreseringsmodi. Har dette nokon innvirknad på ISA definisjonen for denne tenkte maskina? grunngi svaret.

d. Denne maskina er mest sansynleg ei RISC-maskin, kvifor?

OPPGÅVE 4: DATAMASKINER (20% (8% PÅ A OG C; 4% PÅ B))

a. Figur 9 og figur 10 i vedlegg viser forskjelige versjonar av IJVM-mikroarkitektur. Kva grep er gjort og korleis bidrag dei til auka ytelse samanlikna med den opprinlege mikroarkitekturen i figur 5?

b. Kva innvirknad har endringane i mikroarkitekturen gitt i spørsmål a på ISA-nivå?

c. Nytt figur 5, figure 8 og figure 10. Kva kan klokke tida tilnerma reduserast til ved å endre microarkitekturen frå implementasjonen i figur 5 til samlebandsimplementasjonen i figur 10?

OPPGÅVE 5: DIVERSE (20%)

Finn rett svar alternativ for oppgåvene. Korrekte svar gir 4% utteljing, feil svar gir -2% og veit ikkje (ikkje svar, fleire svar) gir inga utteljing.

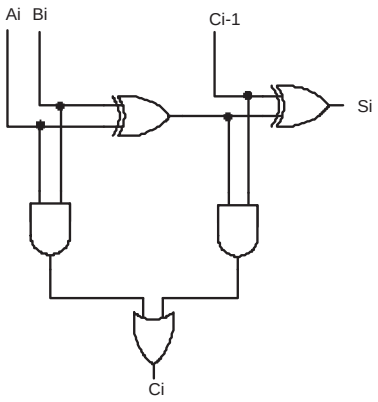
a. Kva ligg i *Addr* felte i *MIR*?, sjå figur 5

   1) Adressa til neste mikroinstruksjon i *control store*.
   2) Inneheld kunn gyldig adresse viss skal gjerast eit betinga hopp, aktivt J-bit.
   3) Til ei kvar tid ein kopi av innhalde i *MBR*.
   4) *control store* start og slutt adresse, for aktiv mikroinstruksjon.

b. Kva påstand er korrekt for ein ein-brikke mulitiprosessorer (CMP) .

   1) Er ein ”Array computer”.
   2) Er type SIMD.
   3) Er av type homogen eller hetrogen.
   4) Er av type MIMD og nyttar alltid ein ”crossbar” for komunikasjon mellom prosessorkjernane.

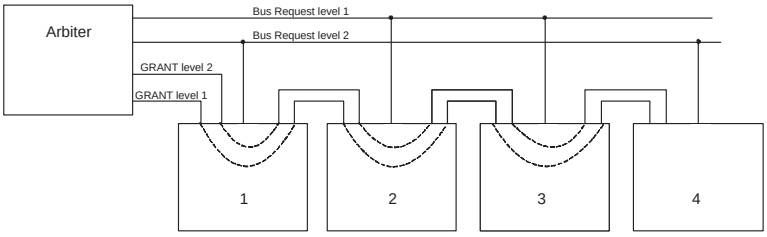
c. Kva er skisert i figur 3?



Figur 3: Mystisk dings.

1) Fulladder.
2) Halvadder.
3) Statisk RAM-celle.
4) Multipleksar.

d. Korleis er einingane i figur 4 prioritert? Nivå 1 (level 1) er prioritert over nivå 2 (level 2). Rekkefølgja er gitt frå høgaste til lågaste.



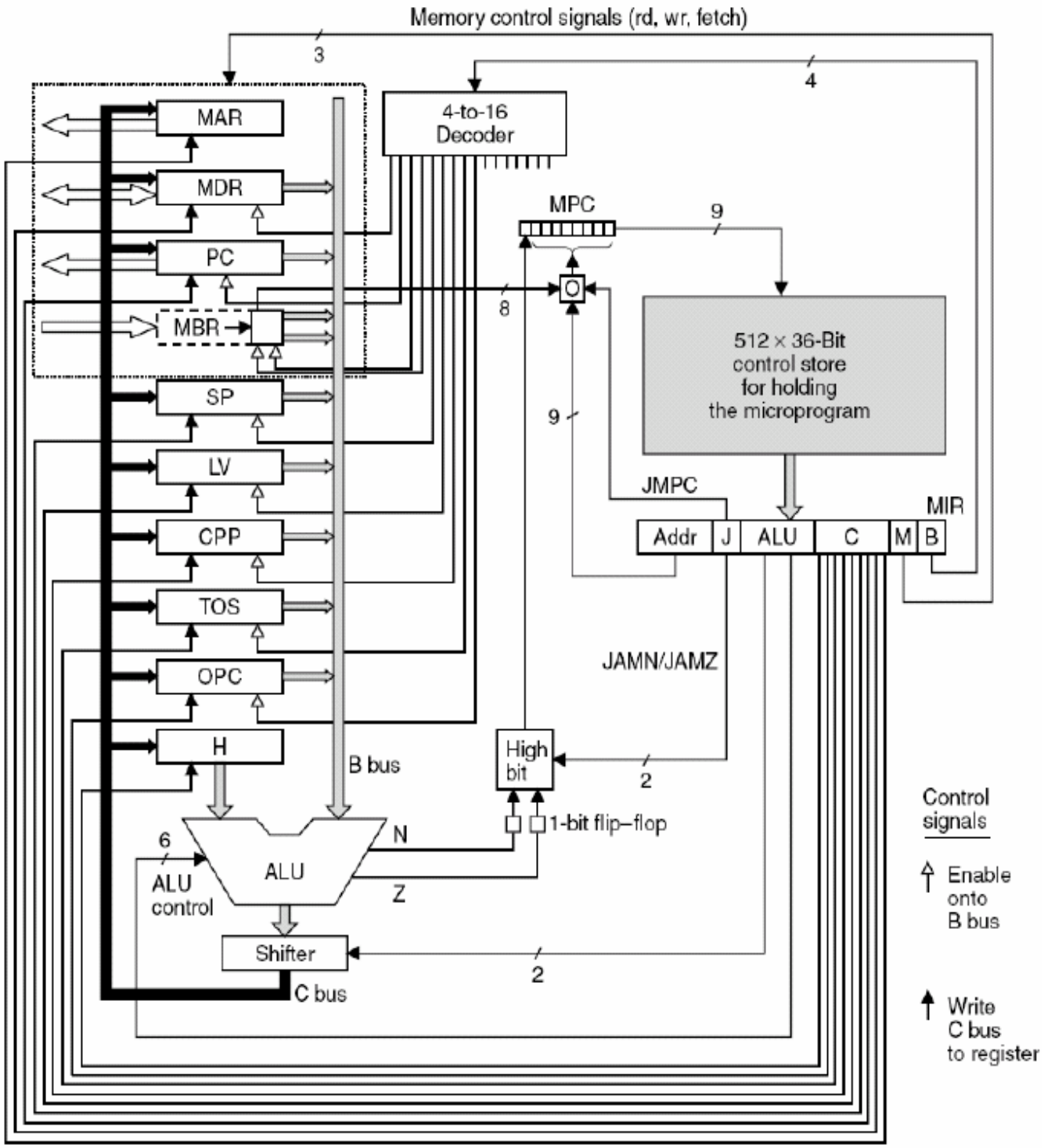
Figur 4: Sentralisert bussarbitrering.

1) 1, 2, 3, 4
2) 2, 3, 1, 4.
3) 1, 4, 2, 3.
4) 3, 2, 4, 1.

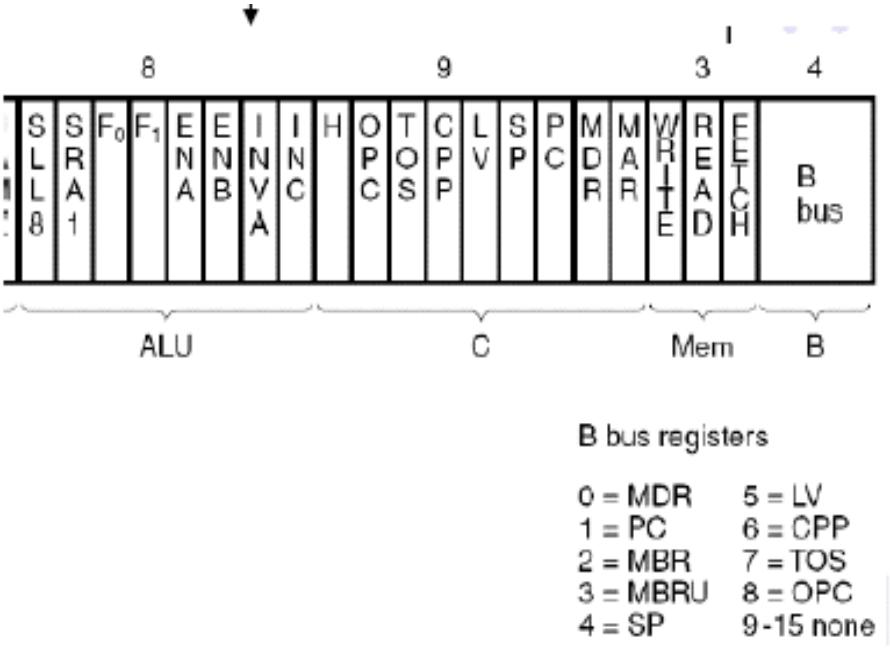
e. Under følgjer ein rekke påstandar om datamaskinkomponentar. Kva påstand er korrekt?

1) Asynkrone bussoverføring må nytte ei global klokke.
2) Ein ALU som kan utføre addisjon og invertere kan utføre subtraksjon.
3) EPROM og flash-minne har ekvivalent funksjonalitet.
4) EEPROM og flash-minne har ekvivalent funksjonalitet.

IJVM vedlegg



Figur 5: Blokkdiagram (IJVM).



Figur 6: Mikroinstruksjonsformat (IJVM).

Answer Key for The Exam

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

a. I Figur 1 er EPROM, RAM og ein lysdiode (LED) kobla til ein felles buss. Finn adresseområde for einingane. EPROM og RAM har aktivt lågt (logisk ”0”) CS (Chip Select) signal.

Answer: Ved å bruke logikken som adresse dekoding finn ein følgjande adresse rom:
EPROM: hex(0000) - hex(3FFF) (16kB med EPROM)
RAM (1): hex(4000) - hex (7FFF) (16kB med RAM i modul 1)
RAM (2): hex(8000) - hex(BFFF) (16kB med RAM i modul 2, 32kB med RAM totalt)
LED: hex(FFFF) (kunn ein adresse lokasjon er brukt til å adressere LED)

Eit alternativt svar som også vert rekna som korrekt er:
EPROM: hex(0000) - hex(3FFF)
RAM: hex(4000) - hex (BFFF)
LED: hex(FFFF)
Då reknar ein RAM som eit område på 32kB.

b. Er det mogleg å utvide tilgjengeleg RAM med med 16kB (16384 bytes)? Grunngi svaret.

Answer: Her er det to svar som gir korrekt svar:
1: Nei det er ikkje plass til 16kB. Det er kunn ledig plass frå adresse C000 til FFFE. Dette gir eit ledigt adresseområde på 16383 byte. Dette er ein byte forlite (LED okkuperar siste adresse (FFFF).

2: Ja det er mulig å utvide med 16kB, **MEN** då vil LED og RAM adresse overlappe ved adresering av adresse FFFF. Denne overlappinga har ikkje noko å seie for RAM aksess sidan LED-en ikkje er kobla til databussen. For LED vil dette gjere at han ogs[lyser når ein skriv til RAM-adresse FFFF

Pass på her, svaret vert ikkje rekna som korrekt viss ikkje ein forklarar.

c. Når lyser lysdioden (LED)?

Answer: Når ein SKRIV til adresse hex(FFFF). Alle adresse linjene må vere høge og R/W-signalet må vere høgt, altså W (write)

OPPGÅVE 2: MIKROARKITEKTUR OG MIKROINSTRUKSJONAR (20% (5% PÅ A OG B; 10% PÅ C))

Bruk vedlagte diagram i figur 5, figur 6, figur 7 og figur 8 for IJVM til å løyse oppgåvene.

a. Forklar funksjonen til registeret ”MPC”.

Answer: MicrProgramCounter peikar på microinstruksjon i ”control store”. Ved start av 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.

Ved hoppinstruksjonar vert MPC manipulert for å kunne utføre betingahopp (det eksisterar to alternativ for kva neste mikroinstruksjon er).

Sjå boka for utfølgjande info. Treng berre kva det er og at MPC held orden på kva som er neste microinstruksjon i aktiv instruksjon for max utteljing.

b. Lag mikroinstruksjon(ar) for følgjande IJVM-operasjon: last register ”OPC” med innhaldet i register ”H”.

Sjå vekk frå Addr- og J-felta i mikroinstruksjonsformatet. Angi korrekte bit for ALU, C, Mem og B gitt i Figur 6.

Answer: FIX ALU: 011000 (A) C: 010000000 (OPC) Mem: 000 (ingen mem opprasjon) B: 1111 (15 (non), men alt går i B sidan ALU er satt til kunn A inngang)

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

Sjå vekk frå Addr- og J-feltene i mikroinstruksjonsformatet. Angi korrekte bit for ALU, C, Mem og B gitt i Figur 6.

Answer: 1: Laste LV eller OPC (eventuelt med + 1 option viss ikkje i neste microopp.) inn i H
ALU: 010100 (B) C: 100000000 (H) Mem: 000 (ingen mem opprasjon) B: 1000 (8 OPC)

2 addere H + LV + 1 skriv til TOS (A + B viss + 1 i forige)(eller H + OPC viss LV i forrige).
ALU: 111101 (A+B +1) C: 001000000 (TOS) Mem: 000 (ingen mem opprasjon) B: 0101 (5 LV)

OBS her er det også andre mulige løysingar (mikrointsekvensar som er korrekt.

OPPGÅVE 3: INSTRUKSJONSSETT ARKITEKTUR (ISA)(20%)

I ein tenkt 16 bit arkitektur har ein berre tre typar instruksjonar, last (load), lagre (store) og flytt / ALU (Move / ALU). Figur 2 viser dei tre instruksjonstypane.

a. Utfrå tilgjengeleg informasjon:

   i) Kva er det maksimale antal instruksjonar denne maskina kan ha? grunngi svaret.

   ii) Kva er det maksimale antal brukar register (user registers) denne maskina kan ha?, grunngi svaret.

   Answer: i) Ser her at instruksjonslengden er fast (16 bit). opCode feltet er på 4 bit. Dette gjer at det er mulig å ha 2^4 altså 16 forskjelige instruksjonar.

   ii) Ser at kvart operandfelt også er 4 bit. Dette gjer at det er mogleg å adressera 2^4 register, altså 16. Sjølv om det er fleire felt for operandar kan ein ikkje ha fleire register sidan alle operandar skal kunne peike på "any user register".

b. Kva type arkitektur er dette instruksjonsformatet for?

   Answer: Dette er ein typisk Load / store arkitektur, sjå boka for detaljar.

c. Viss ein endrar formatet for load / store til å bruke instruksjonsfeltet merka DC til å innehalde ein index og brukar *Ra* som base adresse har ein endra addreseringsmodi. Har dette nokon innvirknad på ISA definisjonen for denne tenkte maskina? grunngi svaret.

   Answer: Ja, nye instruksjonar, ny funksjonalitet. Dette endrar korleis instruksjonar vert tolka slik at det resulterar i to forskjelige ISA som ikkje er kompatible.

d. Denne maskina er mest sansynleg ei RISC-maskin, kvifor?

   Answer: Load / store arkitektur, få korte, liklengde instruksjonar, mange general purpose register, dei fleste instruksjonar kan mest sansynleg gjerast i ein klokke periode.

OPPGÅVE 4: DATAMASKINER (20% (8% PÅ A OG C; 4% PÅ B))

a. Figur 9 og figur 10 i vedlegg viser forskjelige versjonar av IJVM-mikroarkitektur. Kva grep er gjort og korleis bidrag dei til auka ytelse samanlikna med den opprinlege mikroarkitekturen i figur 5?

Answer: Buss må ikkje altid gå gjennom H-reg viss to operandar i instruksjon, ferre microinstruksjonar.

IFU slepp bruke tid (microinst) på å hente instruksjonar.

Pipeline kan minke klokke perioden.

b. Kva innvirknad har endringane i mikroarkitekturen gitt i spørsmål a på ISA-nivå?

Answer: I prinsippe ingen, men viss ein ser på ytelse så vil antal klokke periodar for instruksjonar og den globale klokke frekvensen endre seg for dei forskjelige mikroarkitekturane. For pipline designet kan det også vere nødvendigt å angi kva instruksjonar som ikkje kan utførast etter kvar andre utan nop eller stalling av pipline.
Viss ein svarar ingen er det nok for full utteljing.

c. Nytt figur 5, figure 8 og figure 10. Kva kan klokke tida tilnerma reduserast til ved å endre microarkitekturen frå implementasjonen i figur 5 til samlebandsimplementasjonen i figur 10?

Answer: Viss ein ser på kvar det er lagt inn latch-ar og studerar figure 8 ser ein at ALU tiden, dette er det tregaste trinne. Dette gjer då at ein kan tilnærma setje klokkeperioden til Δy.
Alternativt kan ein seie at $\Delta w + \Delta x$ er det tregaste trinne. Kan då setje dette til klokkeperiodetida.
Viss ein ser litt grovare på det kan ein ut frå figur 10 sjå at no er operasjonar delt i tre deler der latch-ar står som buffer. Ein kan då seie at det er mogleg å redusere til 1/3. Alle desse svara gir full utteljing.

OPPGÅVE 5: DIVERSE (20%)

Finn rett svar alternativ for oppgåvene. Korrekte svar gir 4% utteljing, feil svar gir -2% og veit ikkje (ikkje svar, fleire svar) gir inga utteljing.

(i) Kva ligg i *Addr* felte i *MIR*?, sjå figur 5

1) Adressa til neste mikroinstruksjon i *control store*.

2) Inneheld kunn gyldig adresse viss skal gjerast eit betinga hopp, aktivt J-bit.
3) Til ei kvar tid ein kopi av innhalde i *MBR*.
4) *control store* start og slutt adresse, for aktiv mikroinstruksjon.

Answer: 1

(ii) Kva påstand er korrekt for ein ein-brikke mulitiprosessorer (CMP) .

1) Er ein ”Array computer”.
2) Er type SIMD.
3) Er av type homogen eller hetrogen.
4) Er av type MIMD og nyttar alltid ein ”crossbar” for komunikasjon mellom prosessorkjernane.

Answer: 3

(iii) Kva er skisert i figur 3?

1) Fulladder.
2) Halvadder.
3) Statisk RAM-celle.
4) Multipleksar.

Answer: 1

(iv) Korleis er einingane i figur 4 prioritert? Nivå 1 (level 1) er prioritert over nivå 2 (level 2). Rekkefølgja er gitt frå høgaste til lågaste.

1) 1, 2, 3, 4
2) 2, 3, 1, 4.
3) 1, 4, 2, 3.
4) 3, 2, 4, 1.

Answer: 2

(v) Under følgjer ein rekke påstandar om datamaskinkomponentar. Kva påstand er korrekt?

1) Asynkrone bussoverføring må nytte ei global klokke.
2) Ein ALU som kan utføre addisjon og invertere kan utføre subtraksjon.
3) EPROM og flash-minne har ekvivalent funksjonalitet.
4) EEPROM og flash-minne har ekvivalent funksjonalitet.

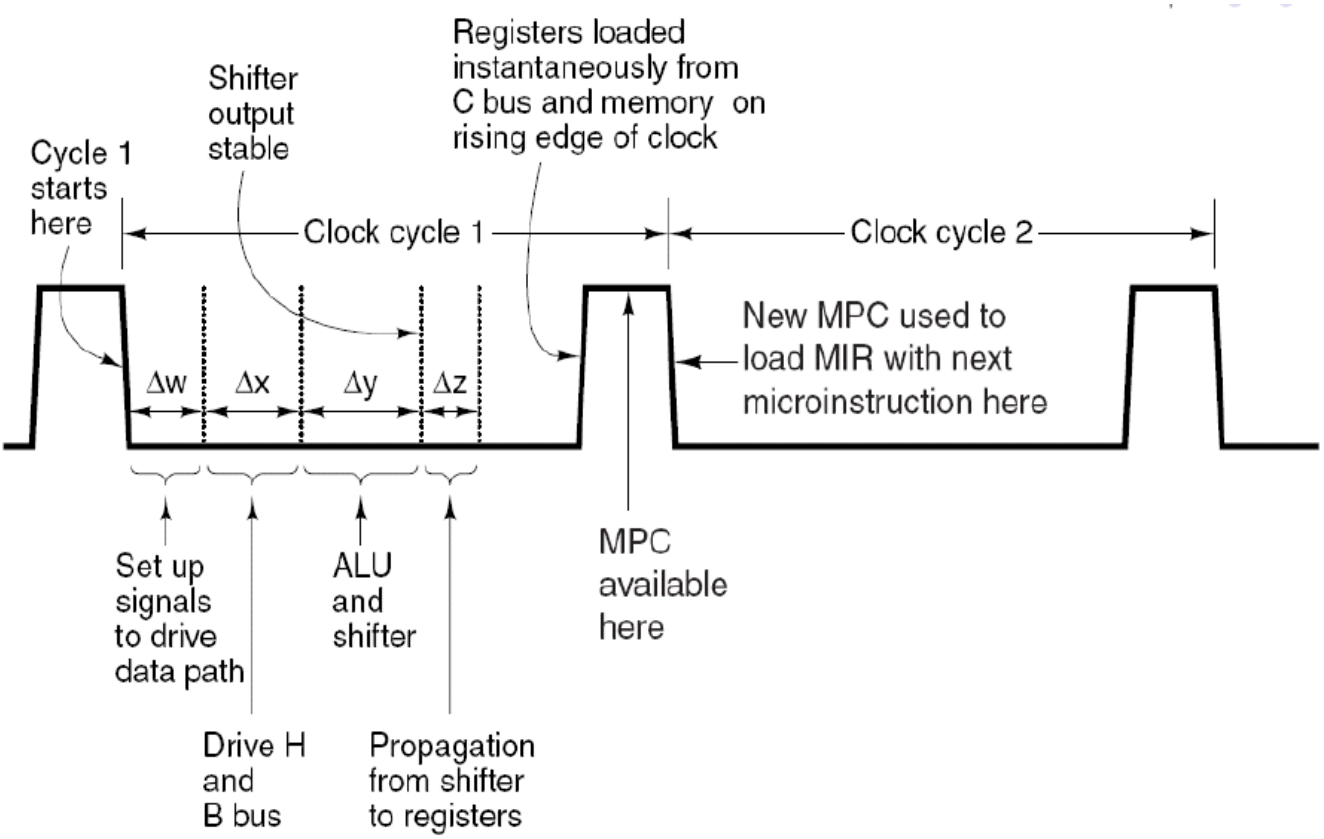
Answer: korrekt? 2

IJVM vedlegg

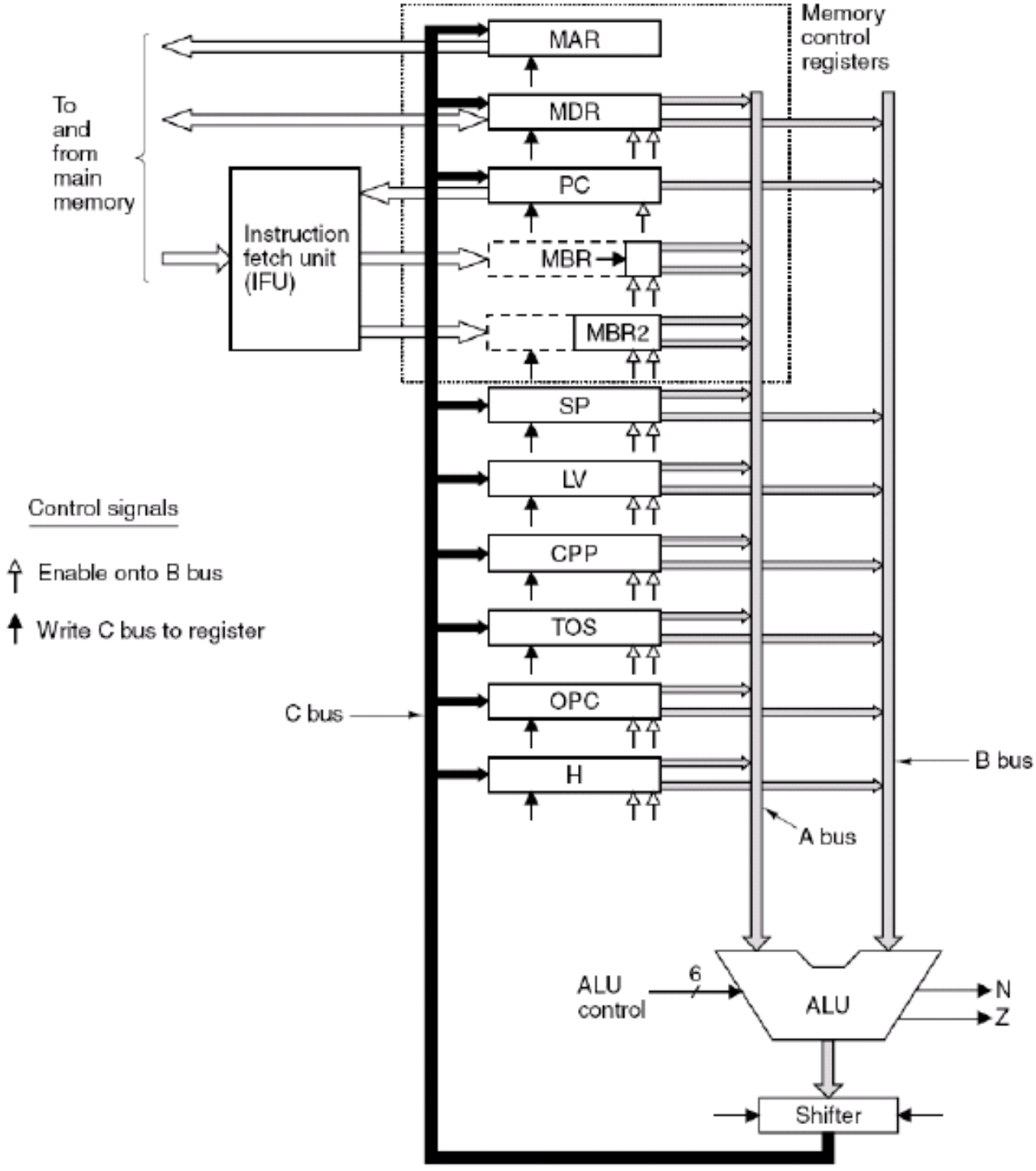
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

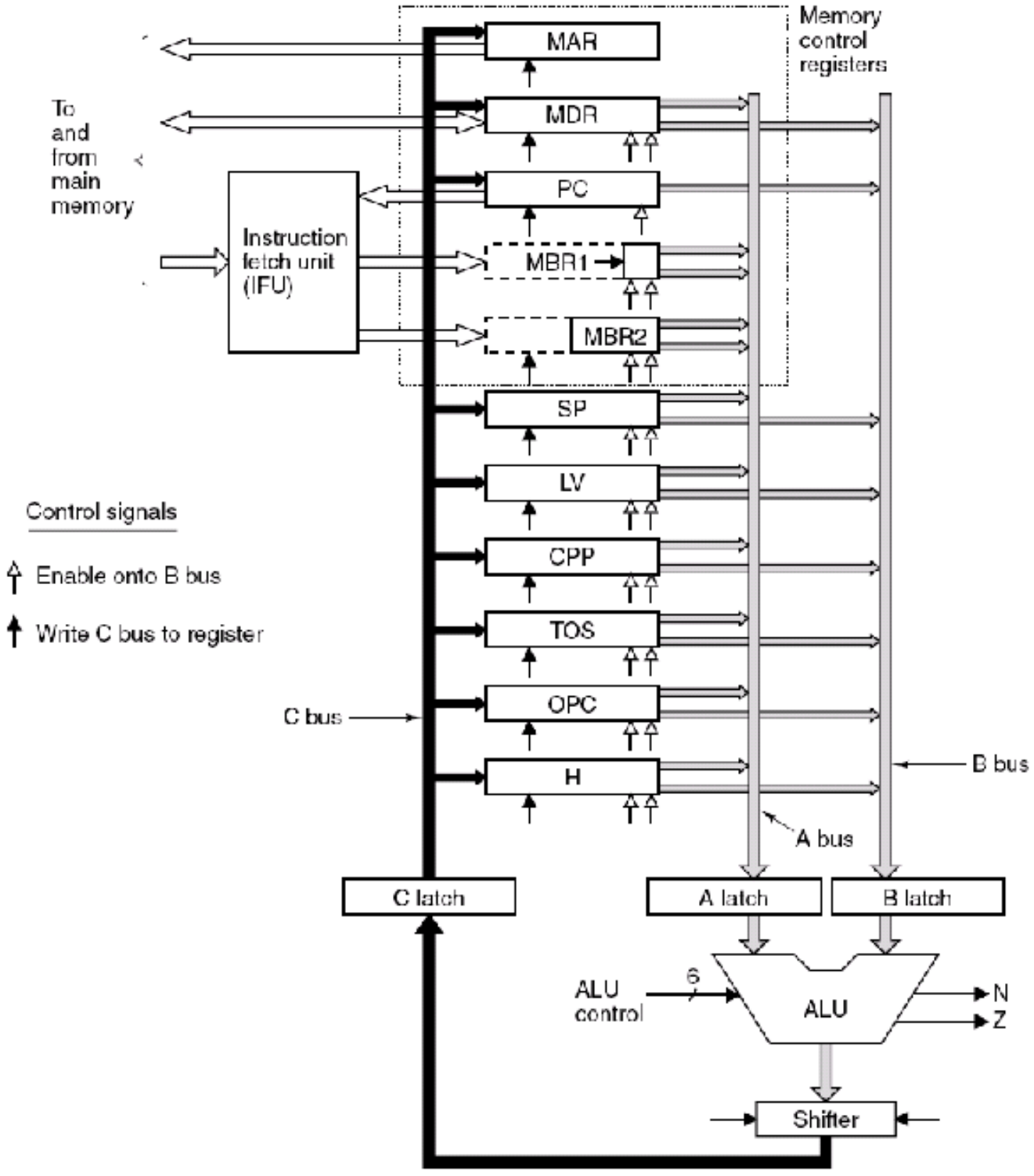
Figur 7: Funksjonstabell for ALU (IJVM).



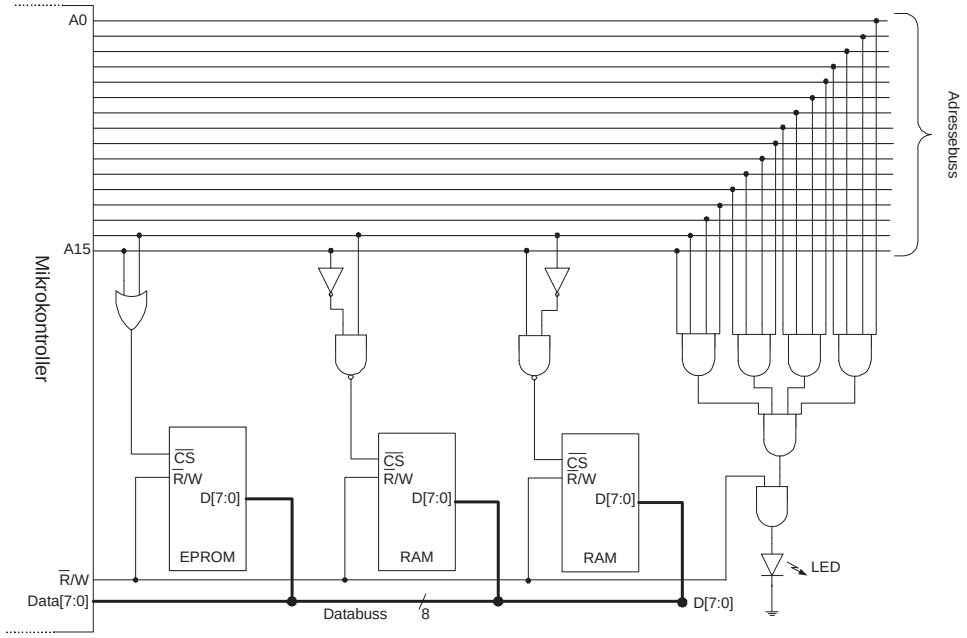
Figur 8: Timingdiagram (IJVM).



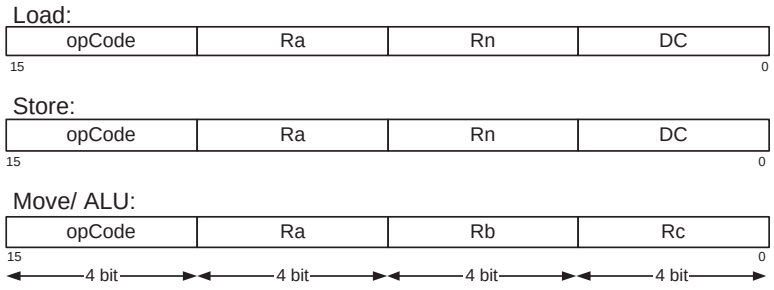
Figur 9: Alternativ mikroarkitektur I.



Figur 10: Alternativ mikroarkitektur II.



Figur 11: Adressedekoding.



Load:

Ra: points to memory location, any user register
Rn: data destination register, any user register
DC: Don't Care, not in use

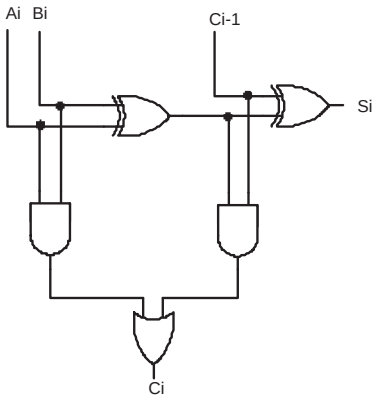
Store:

Ra: points to memory location, any user register
Rn: data source register, any user register
DC: Don't Care, not in use

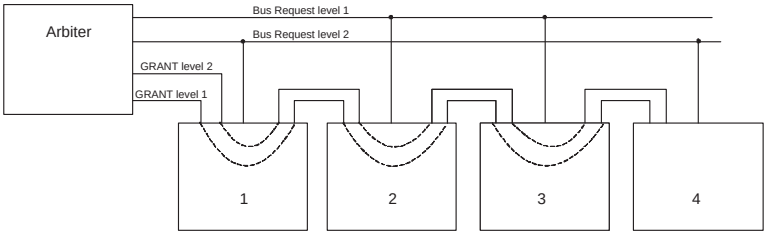
Move/ ALU::

Ra: destination register, any user register
Rb: source register, any user register
Rc: source register, any user register

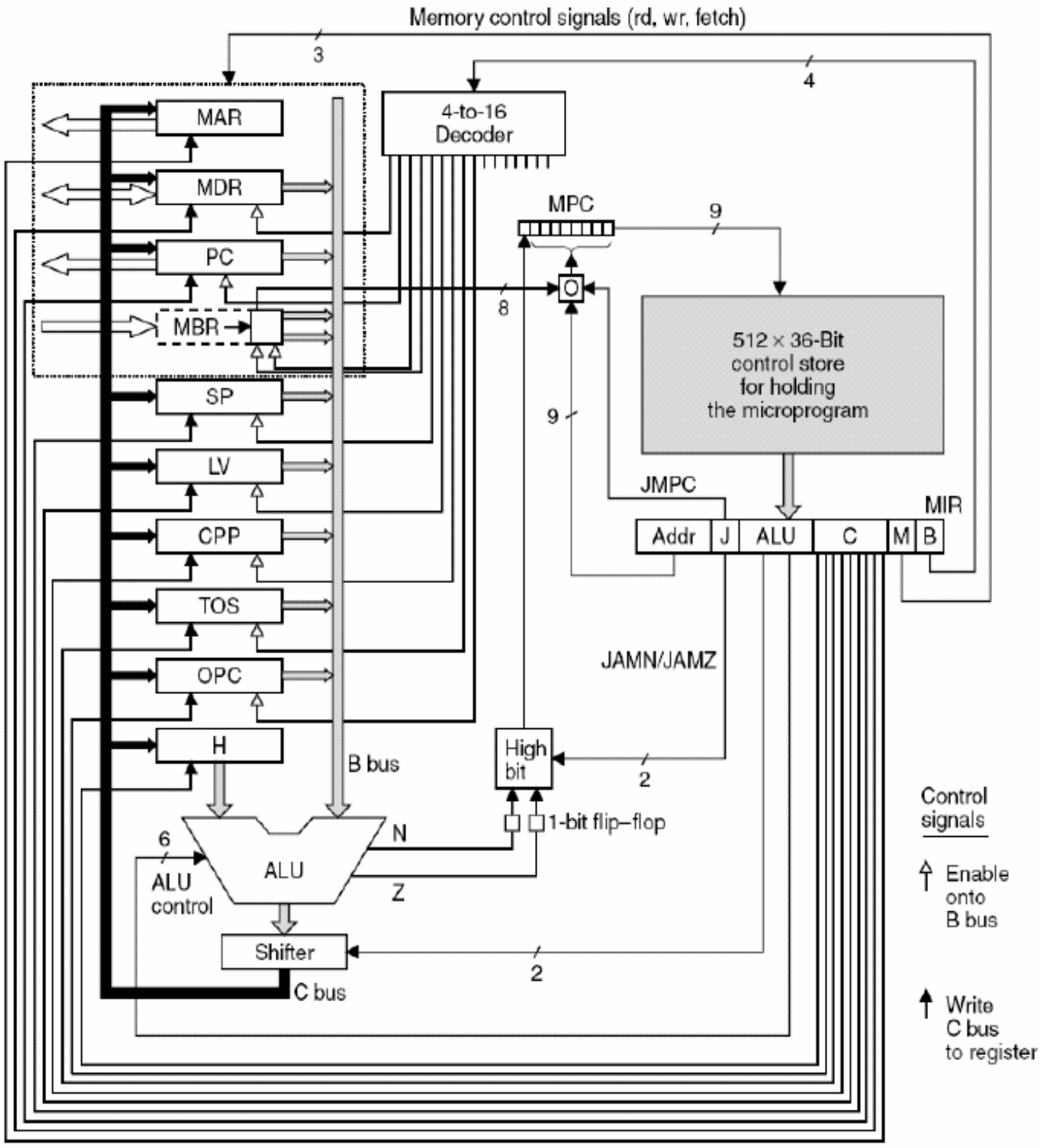
Figur 12: Adressedekoding.



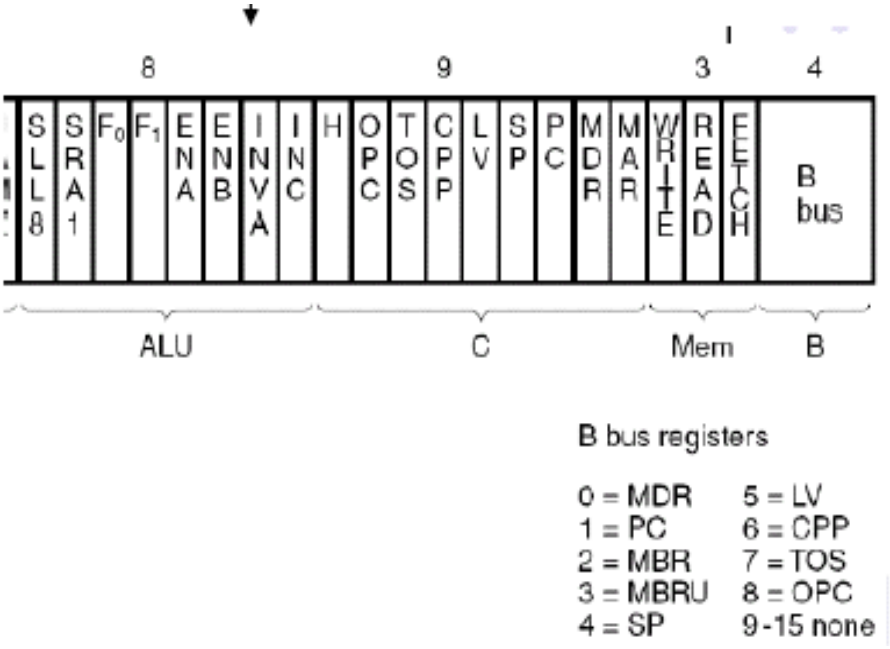
Figur 13: Mystisk dings.



Figur 14: Sentralisert bussarbitrering.



Figur 15: Blokkdiagram (IJVM).

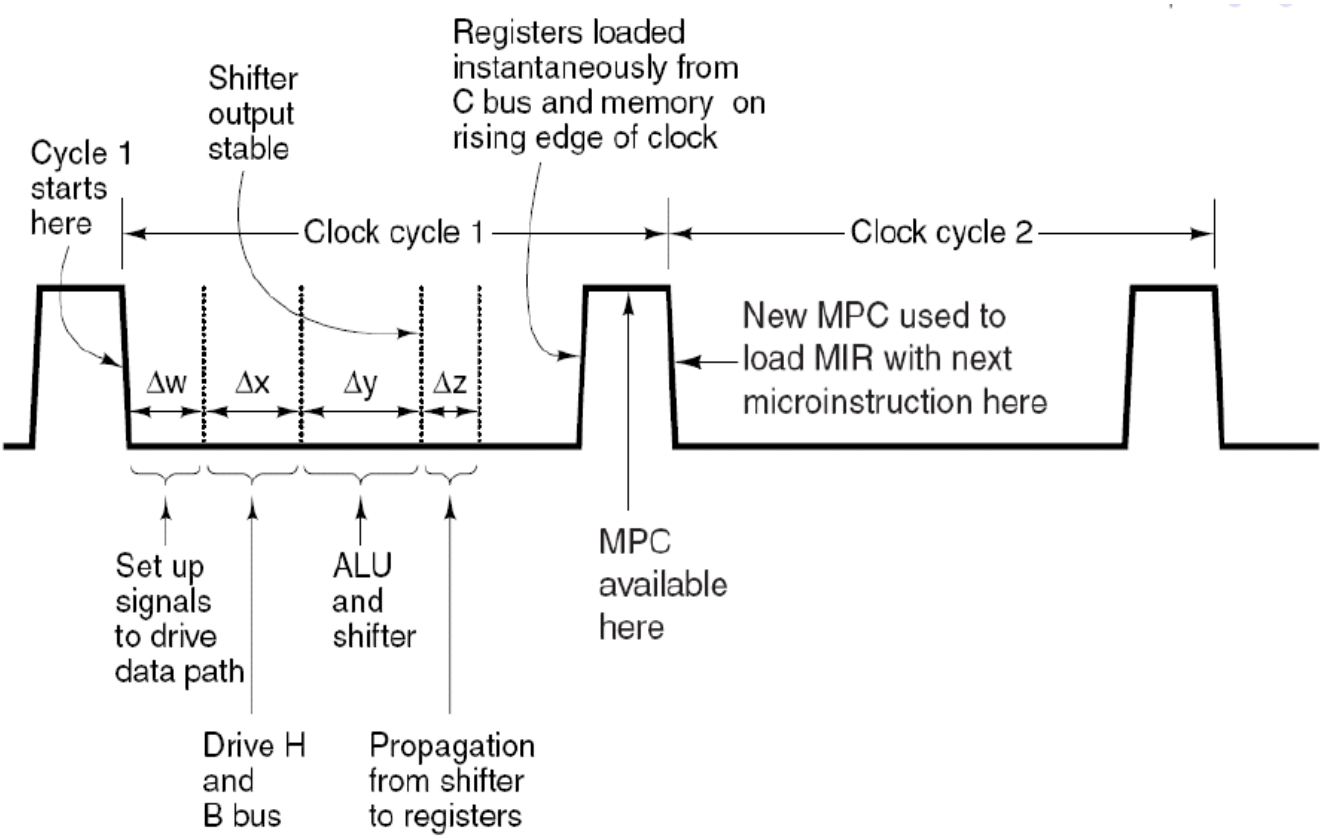


Figur 16: Mikroinstruksjonsformat (IJVM).

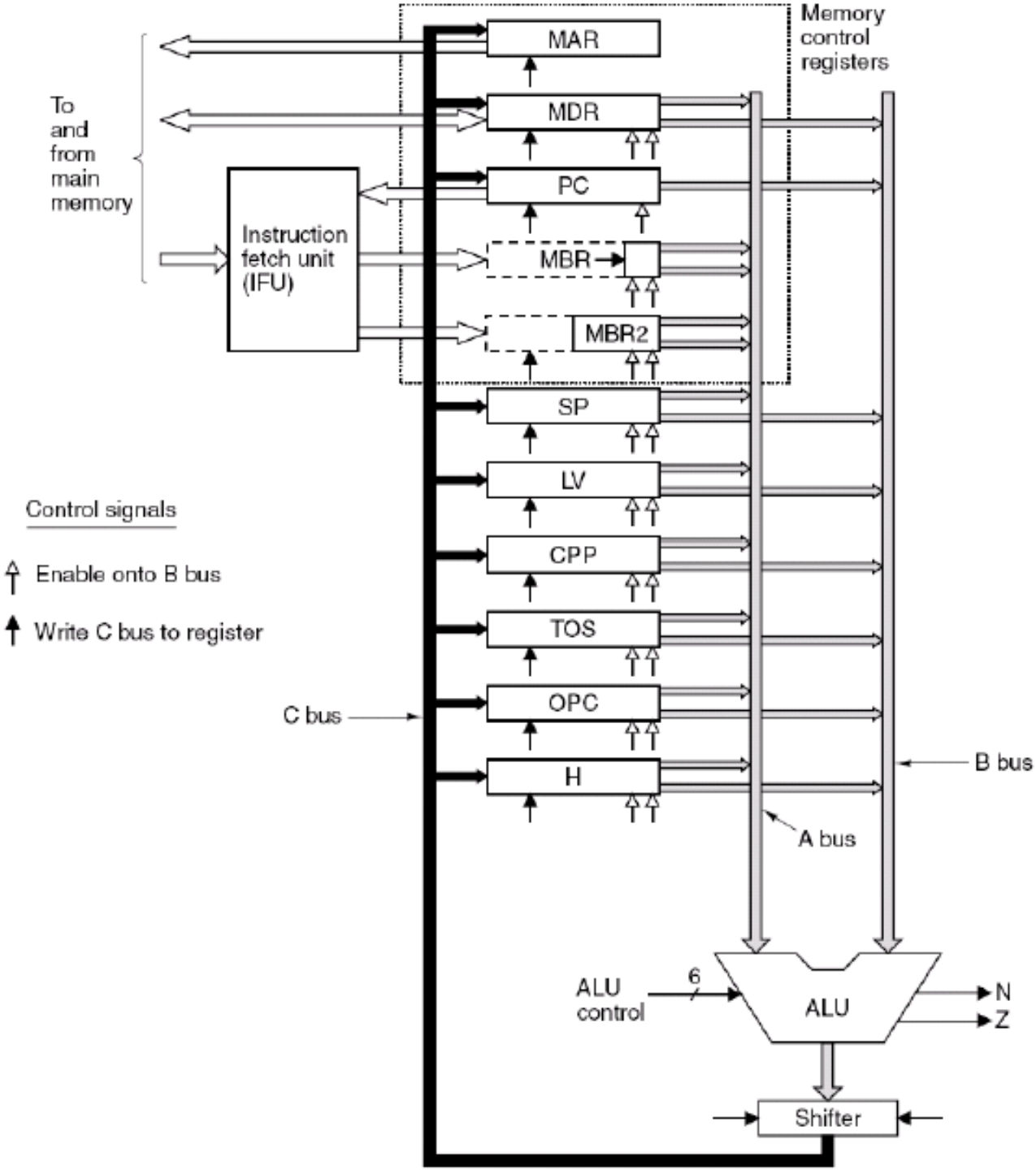
F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	0	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0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

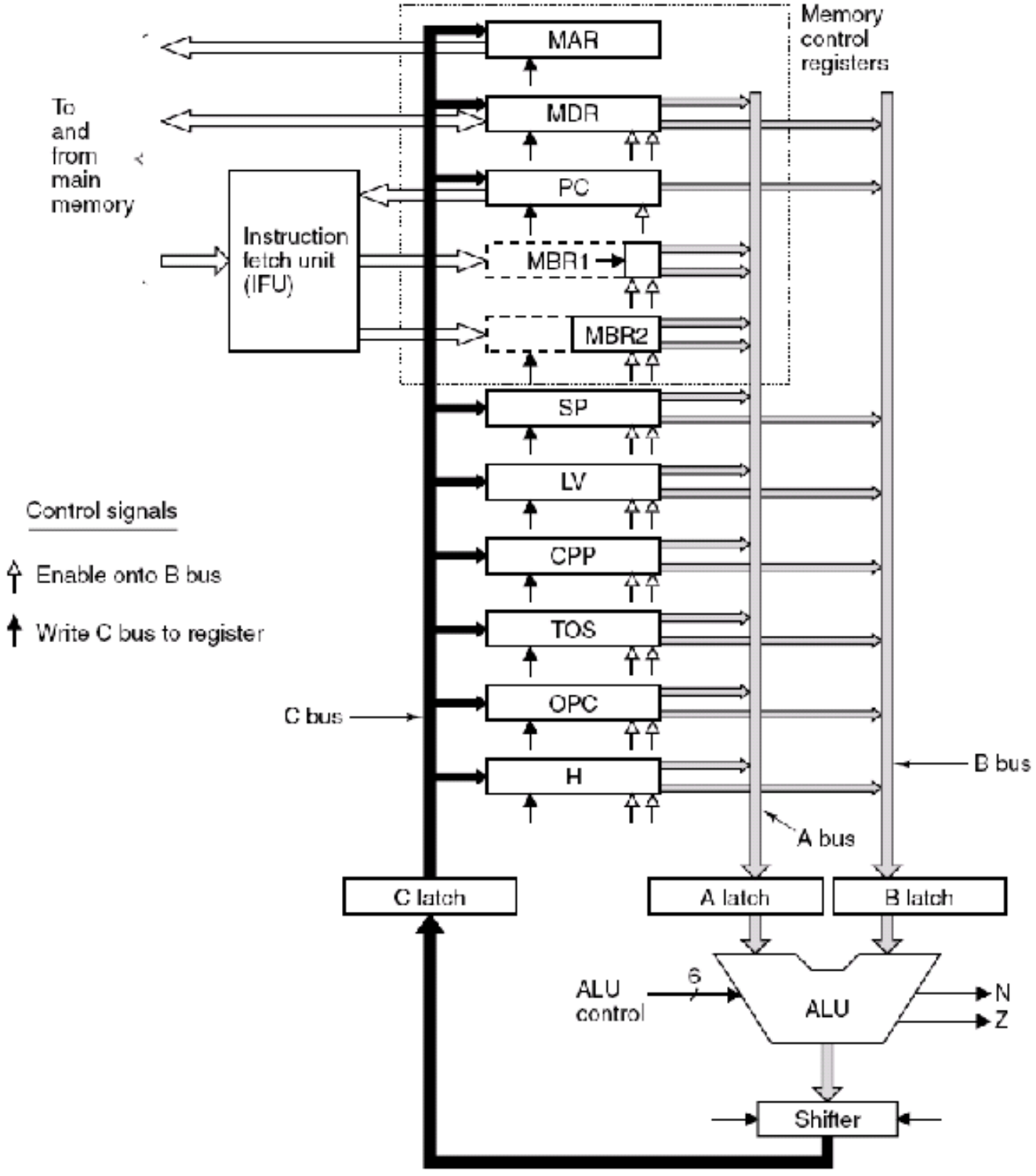
Figur 17: Funksjonstabell for ALU (IJVM).



Figur 18: Timingdiagram (IJVM).



Figur 19: Alternativ mikroarkitektur I.



Figur 20: Alternativ mikroarkitektur II.