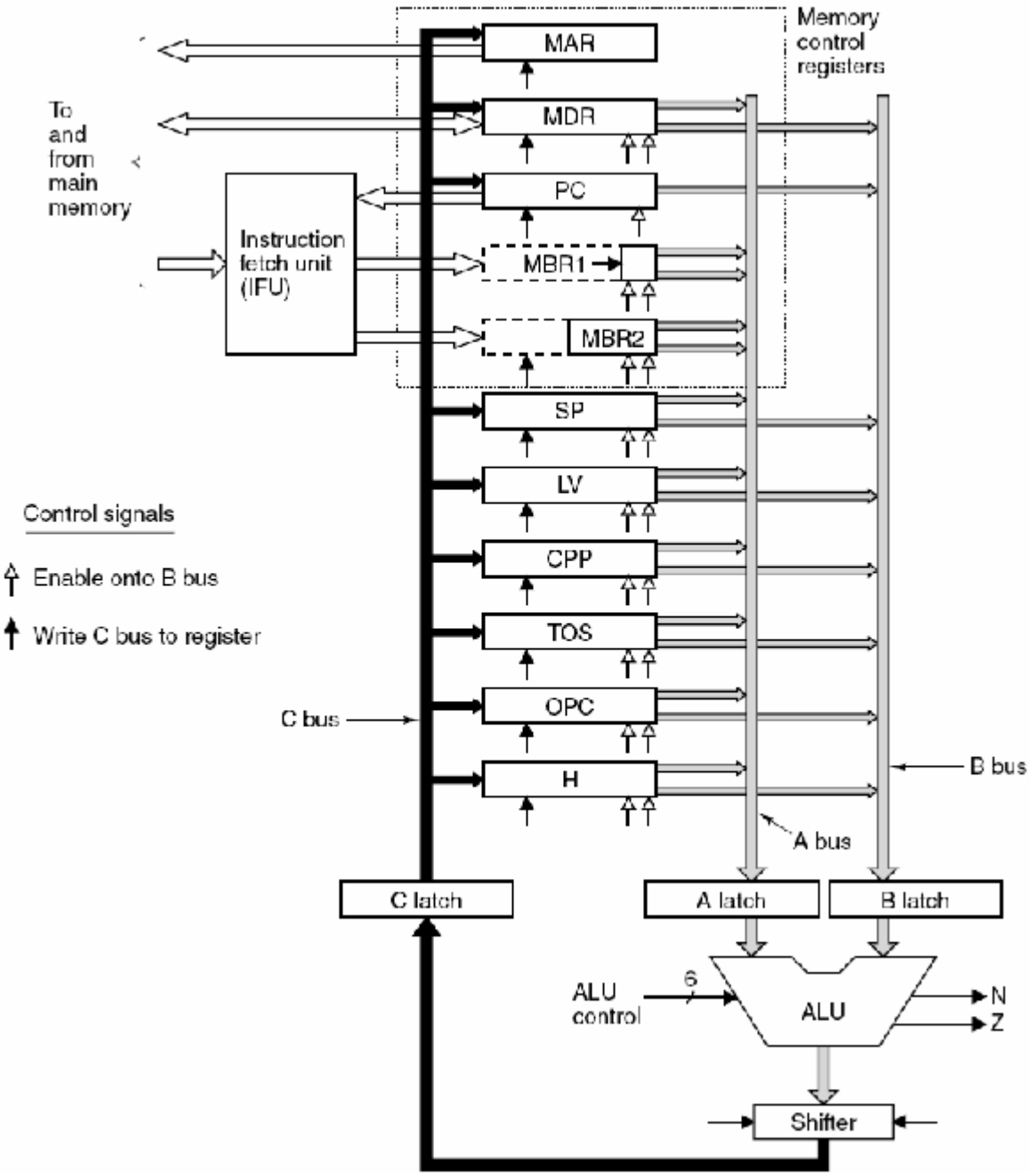
Fortsetelse Microarchitecture level

Før hurtigbuffer, siste pipelineTing

Dataavengighet (pipelineTing)

- Parallellitet
  - Instruksjonsnivåparallellitet
    - Fleire instruksjonar utføres samtidig (1 prosessor)
    - Samleband
      - Eks: Unødvendig at ALU er ledig mens CPU leser instruksjon
    - Superskalaritet: Duplisering av CPU-komponenter
  - Utføring av instruksjonar
    - Resultat av inst x
    - Data for inst Y
  - Har då avengigheit
    - Read after Write (RAW)
    - Write after Read (WAR)
    - Write after Write (WAW)
  - Må passe på
    - Instruksjonar ikkje utførest for tidleg
    - DVS brukarendring data



Dataavengighet SWAP eksempelet

Label	Operations	Comments
swap1	MAR = SP – 1; rd	Read 2nd word from stack; set MAR to SP
swap2	MAR = SP	Prepare to write new 2nd word
swap3	H = MDR; wr	Save new TOS; write 2nd word to stack
swap4	MDR = TOS	Copy old TOS to MDR
swap5	MAR = SP – 1; wr	Write old TOS to 2nd place on stack
swap6	TOS = H; goto (MBR1)	Update TOS

- Mic-2 kode vist over
- Nytt problem med samlebånd: Er verdier klare?
- swap3 leser MDR som blir skrevet pga. rd i swap1
- Med overlapping er det mulig at swap3 leser før verdi blir skrevet ➔ FEIL!
- Les-etter-skriv eller sann dataavhengighet

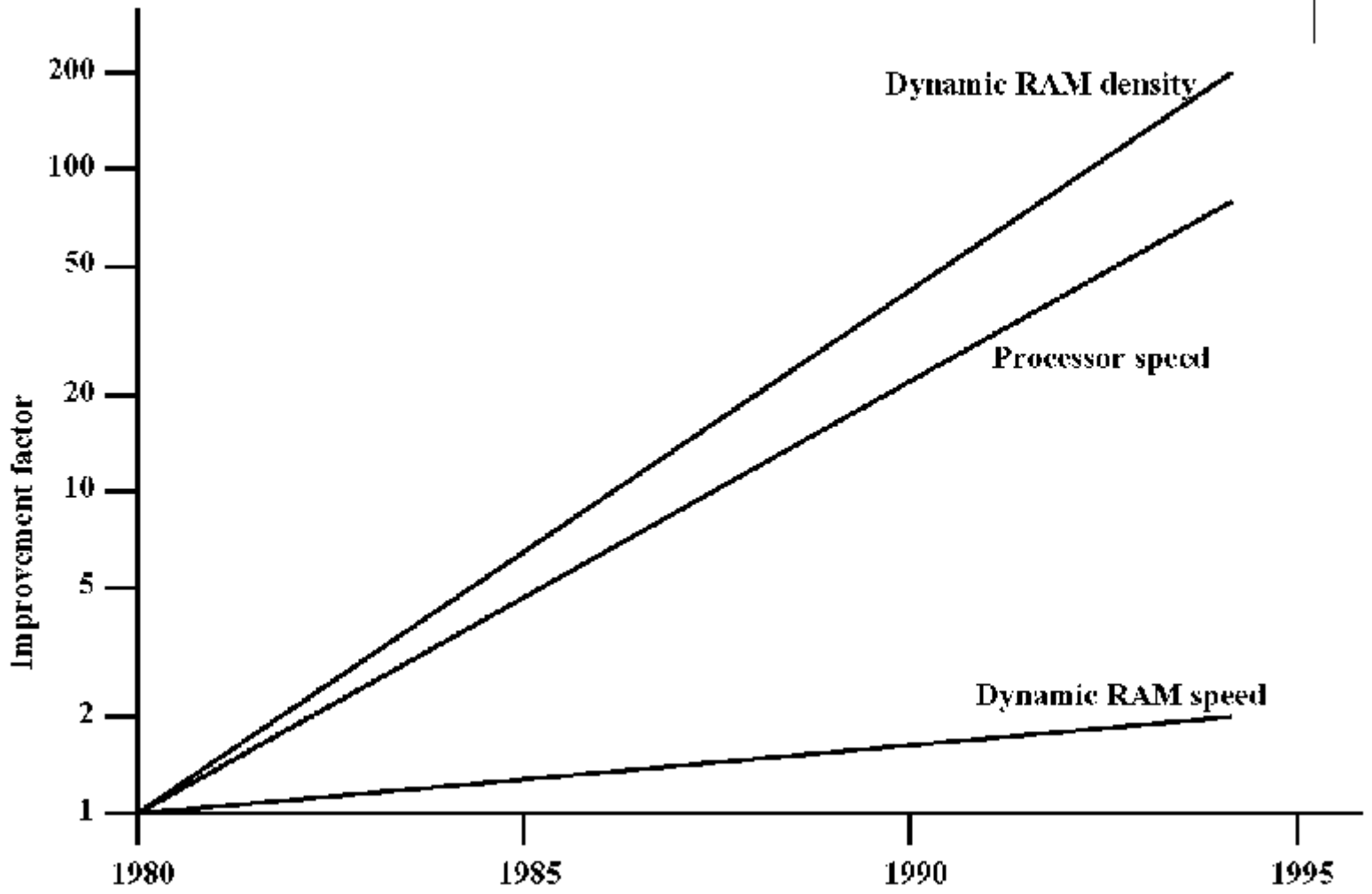
Dataavengighet SWAP eksempelet

	Swap1	Swap2	Swap3	Swap4	Swap5	Swap6
Cy	MAR=SP–1;rd	MAR=SP	H=MDR;wr	MDR=TOS	MAR=SP–1;wr	TOS=H;goto (MBR1)
1	B=SP					
2	C=B–1	B=SP				
3	MAR=C; rd	C=B				
4	MDR=Mem	MAR=C				
5			B=MDR			
6			C=B	B=TOS		
7			H=C; wr	C=B	B=SP	
8			Mem=MDR	MDR=C	C=B–1	B=H
9					MAR=C; wr	C=B
10					Mem=MDR	TOS=C
11						goto (MBR1)

Ser at Swap3 ikke kan starte i klokkesyklus 3
Veldig viktig å passe på!

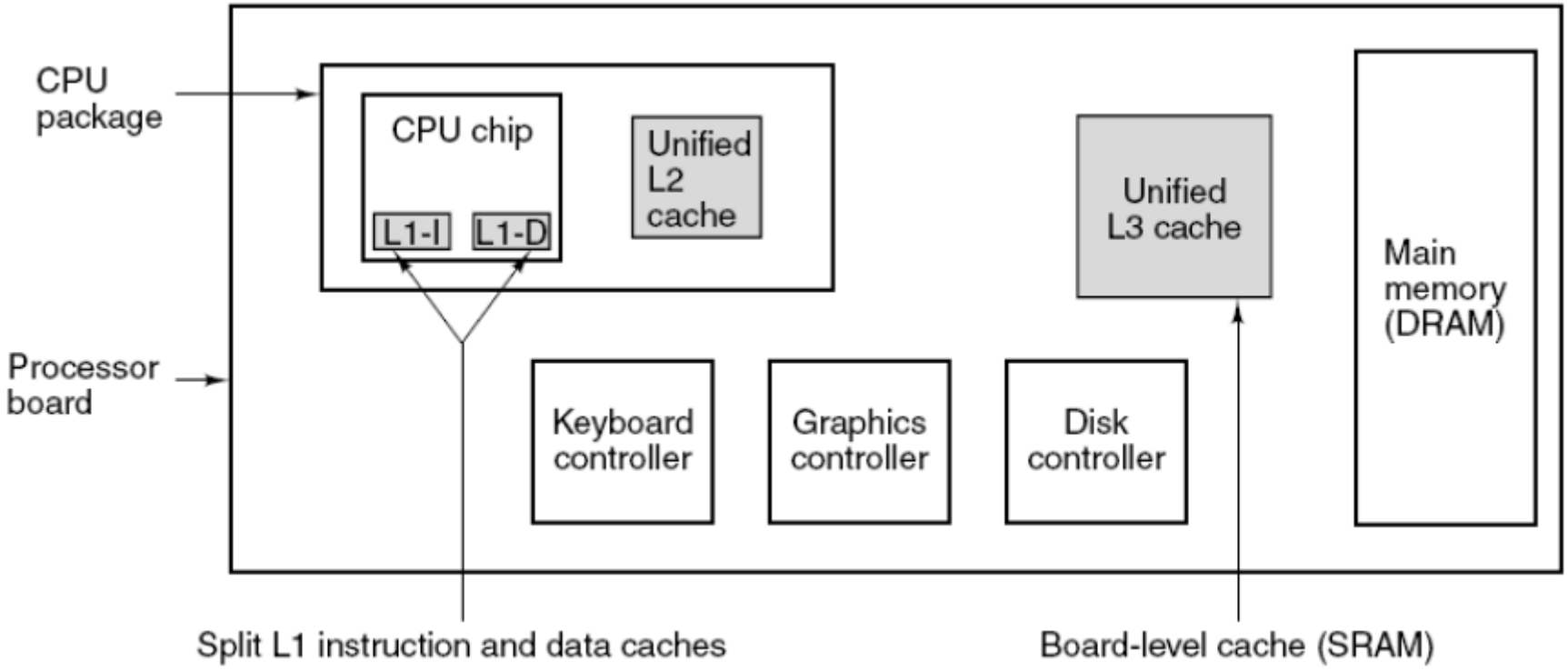
NTNU
Innovation and Creativity

Hurtigbuffer

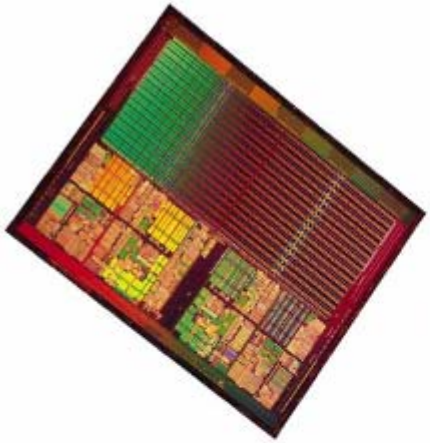


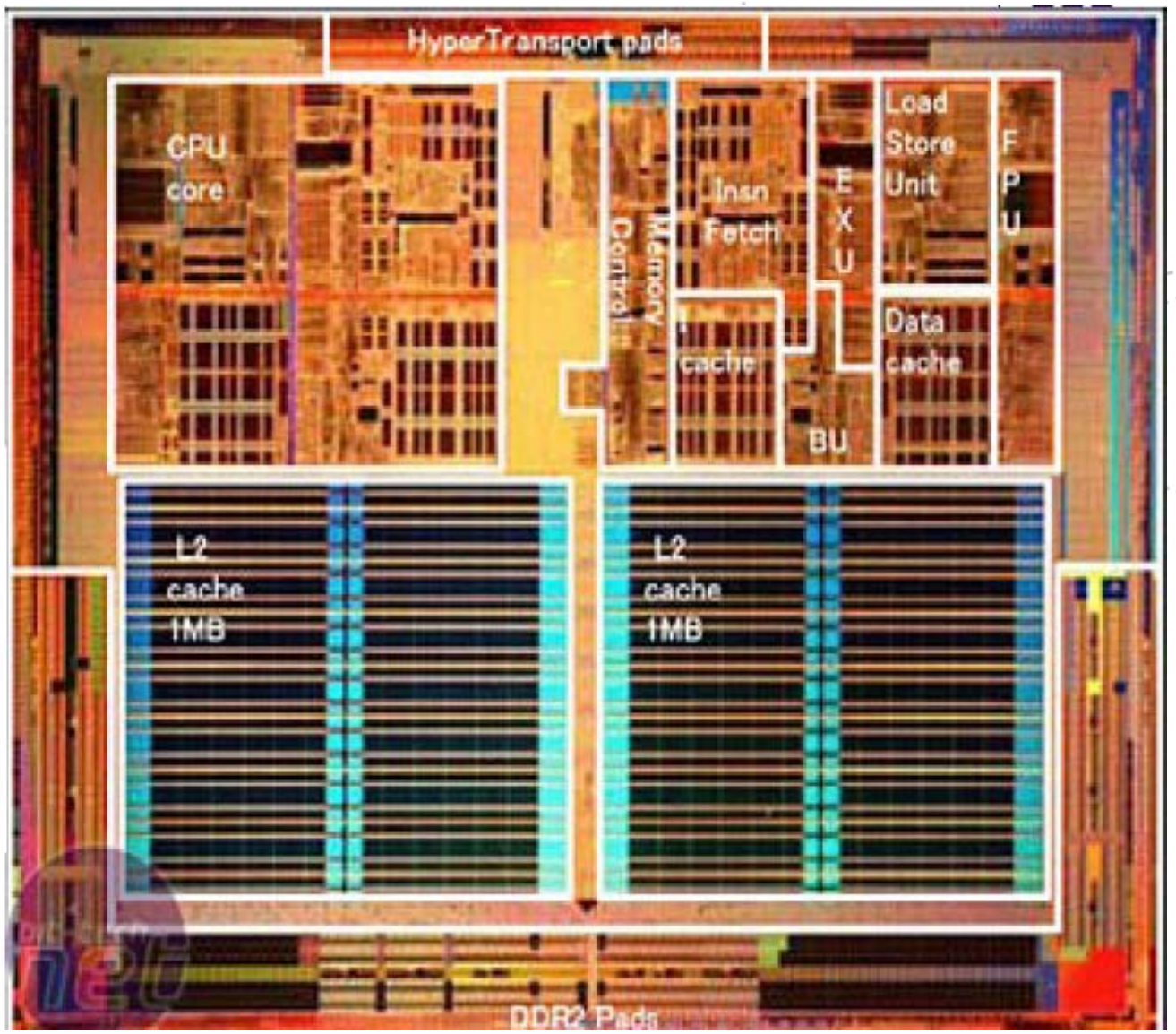
NTNU
Innovation and Creativity

Hurtigbuffer



Hurtigbuffer





Hurtigbuffer

Lokalitetsprinsippet (1/2)

- Hurtigbuffer
  - Størrelse gjerne 0,1% av hovedlager
  - Likevel har hurtigbuffer ofte 95% treffrate
- Hvordan er dette mulig?
  - Lager aksesseres **ikke** i et tilfeldig mønster
  - Kan forutse fremtidige lageraksesser
- Lokalitet i rom:
  - Henter vi fra adresse 100 er det sannsynlig at vi snart kommer til å hente fra adresse 101
    - Instruksjoner: Sekvensiell utførelse
    - Data: Ofte lagret i blokker (tabeller, bilder o.l.)

NTNU
Innovation and Creativity

- 1 ILOAD
- 2 PC = PC + 1
- 3 IADD
- 4 PC = PC + 1
- 5 ISTORE
- 6 PC = PC + 1
- 7 ILOAD
- 8 PC = PC + 1
- 9 BIPUSH
- 10 PC = PC + 1
- 11 IF_ICMPEQ (betinga hopp) **L1**
- 12 PC = Opdater PC
- 13 ILOAD
- 14 PC = PC + 1
- 15 BIPUSH
- 16 PC = PC + 1
- 17 ISUB
- 18 PC = PC + 1
- 19 ISTORE
- 20 PC = PC + 1
- 21 GOTO **L2**
- 22 PC = **L2**
- 23 BIPUSH **L1**
- 24 PC = PC + 1
- 25 ISORE
- 26 PC = PC + 1
- 27 XXXXX **L2**

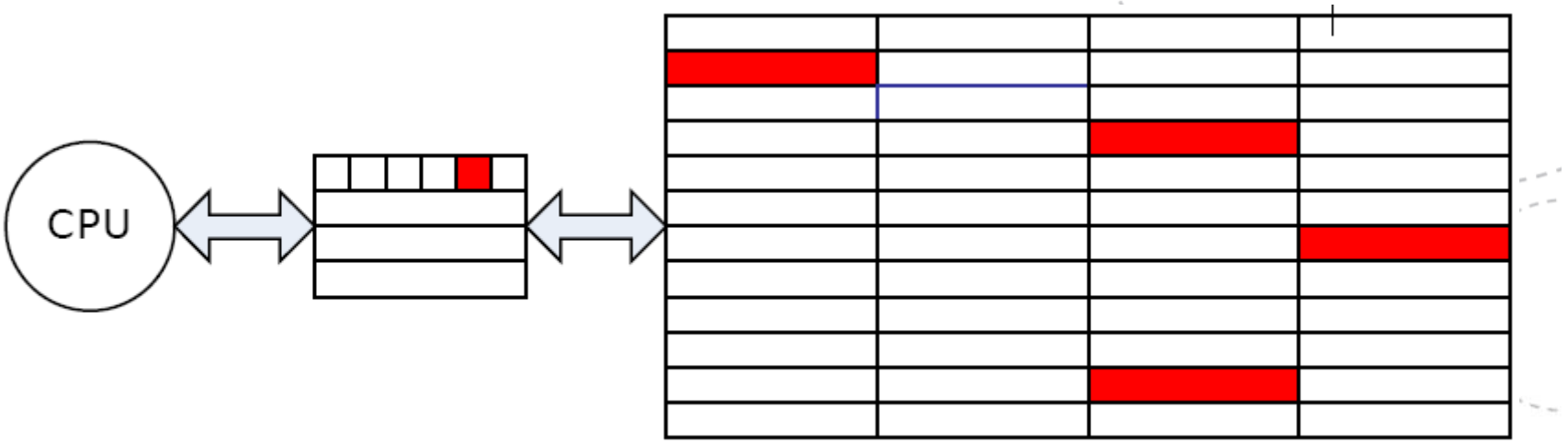
NTNU
Innovation and Creativity

Hurtigbuffer

Lokalitetsprinsippet (2/2)

- Lokalitet i tid:
  - Henter vi fra adresse 100 nå, er det sannsynlig snart vil lese fra samme adresse
    - Instruksjoner: Løkker, subrutiner, ...
    - Data: Ofte brukte variabler, ...
- Dermed: Hentes noe fra hovedlager...
  - Legg det i hurtigbuffer
  - Hent også data fra etterfølgende adresse(r) til hurtigbufferet
- ➔ Sannsynlig at det vi trenger er i hurtigbuffer

Hurtigbuffer organisering



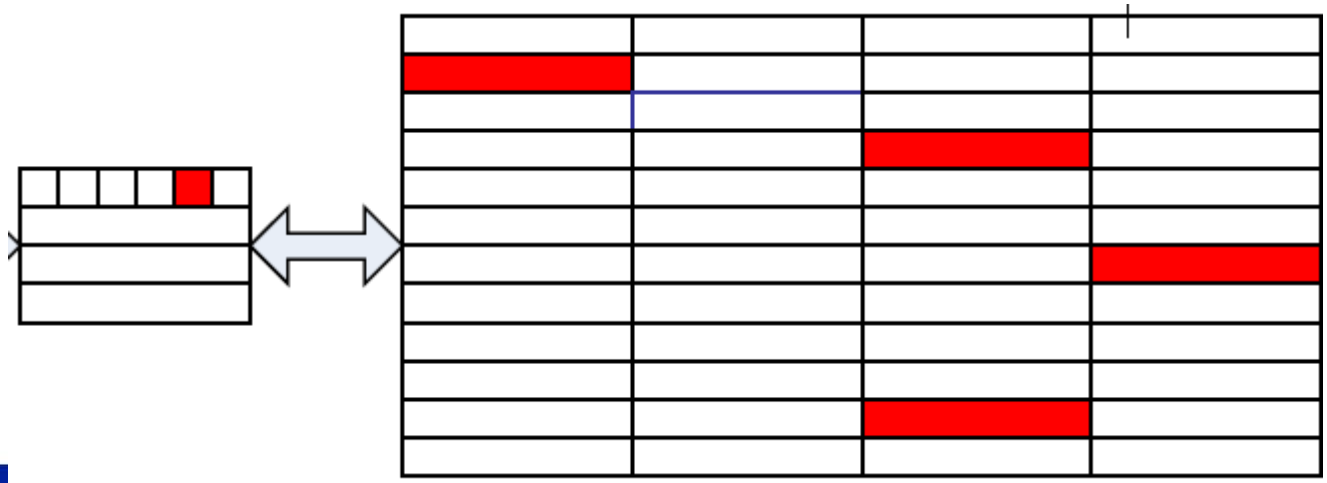
- Hovedlager deles inn i linjer (eks: á 64 bytes)
- Hurtigbuffer inneholder kopi av noen av linjene
- Overføring av data blir dermed:
  - Mellom prosessor og hurtigbuffer: Byte/Ord
  - Mellom hurtigbuffer og hovedlager: Linjer

Hurtigbuffer avbilding

- Hurtigbuffer inneholder:
  - Kopi av data fra hovedlager
  - (Del av) hovedlageradresse til hver linje
- Når prosessor aksesserer hovedlager
  - Sjekk om data finnes i hurtigbuffer (match på adr.)
- Viktige problemstillinger:
  - Hvordan sjekker man om noe ligger i hurtigb.?
  - Hvor i hurtigbuffer kan nye linjer plasseres?
  - Hva skal kastes ut hvis hurtigbuffer er fullt?
  - ➔ Avhenger av valgt avbildningsmetode

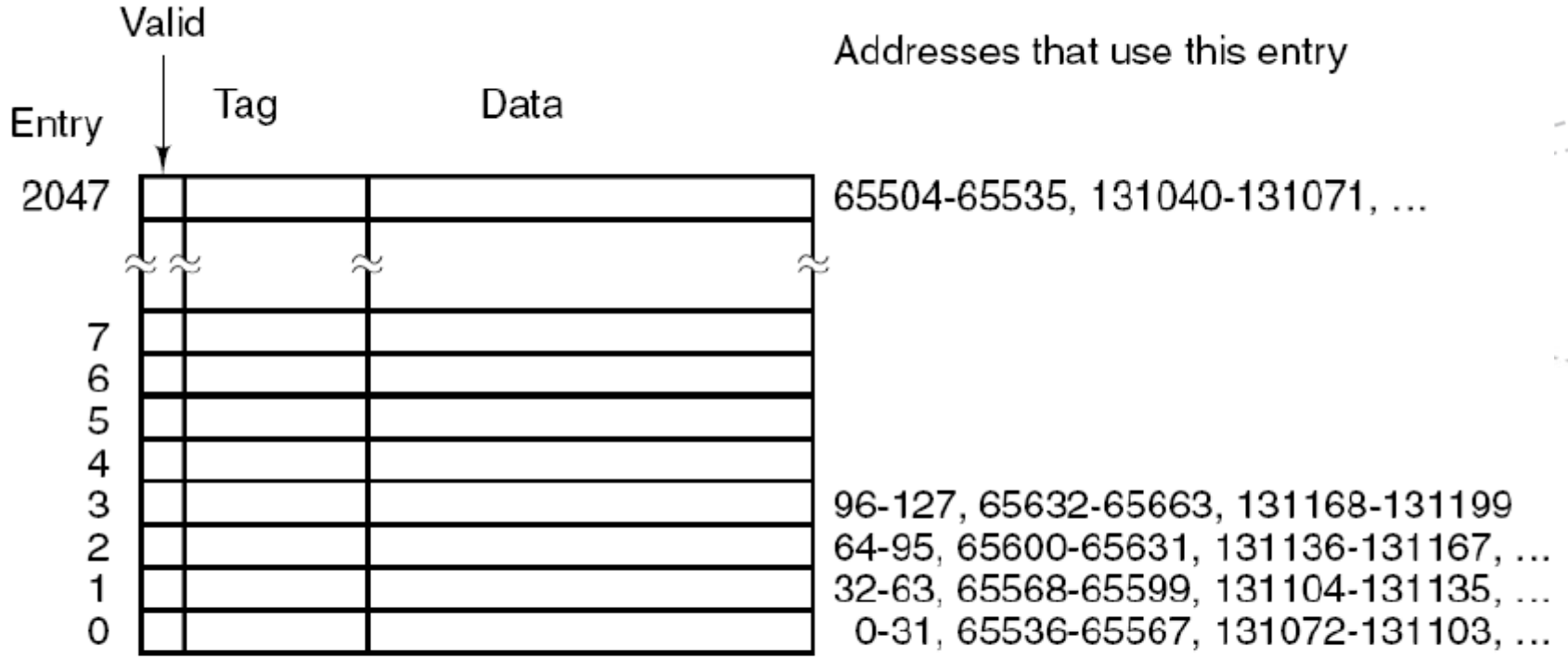
Hurtigbuffer avbilding

- Hurtigbuffer inneholder:
  - Kopi av data fra hovedlager
  - (Del av) hovedlageradresse til hver linje
- Når prosessor aksesserer hovedlager
  - Sjekk om data finnes i hurtigbuffer (match på adr.)
- Viktige problemstillinger:
  - Hvordan sjekker man om noe ligger i hurtigb.?
  - Hvor i hurtigbuffer kan nye linjer plasseres?
  - Hva skal kastes ut hvis hurtigbuffer er fullt?
  - ➔ Avhenger av valgt avbildningsmetode



Hurtigbuffer avbilding

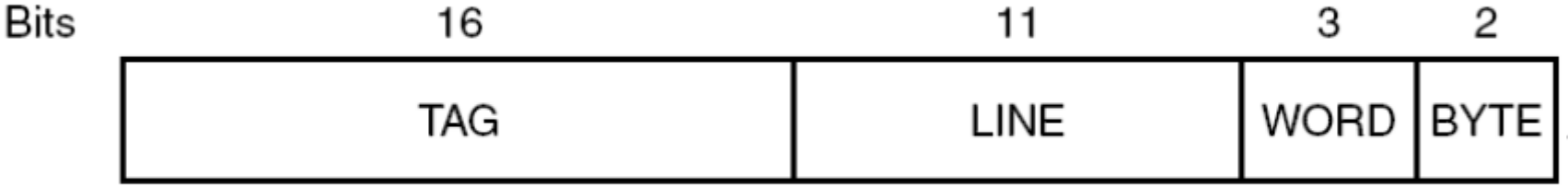
Direkte avbildning (1/3)



- En gitt hovedlagerlinje kan bare ligge på en plass
- Flere hovedlagerlinjer kan ligge på samme plass
  - Tag (”Merkelapp”) forteller hvilken av disse det er

Hurtigbuffer avbilding

Direkte avbildning (2/3)



- Hovedlageradresse deles inn i 4 felt
- Eksempel:
  - Maks 4 GB hovedlager ➔ 32 bits adresse
  - 32 bits ord ➔ 2 bit BYTE-felt (byte innen ord)
  - 32 bytes linjer ➔ 3 bits WORD-felt (ord innen linje)
  - 2048 hurtigbufferlinjer ➔ 11 bits LINE-felt
  - Resten (32-2-3-11=16) brukes til TAG-felt
- Data som ligger på adresser med like LINE-felt, havner på samme hurtigbufferlinje

Hurtigbuffer avbilding

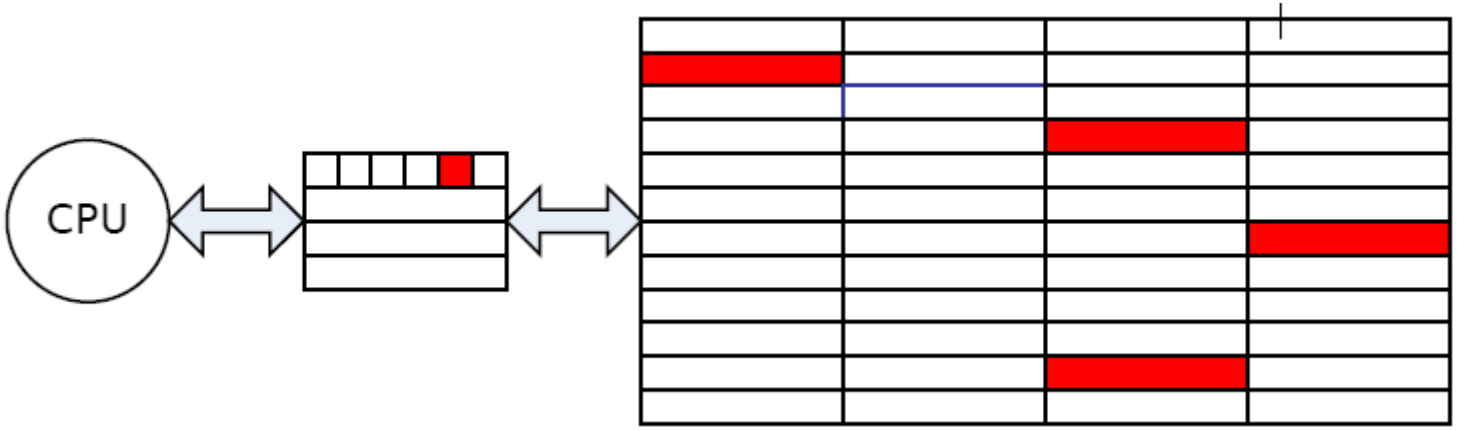
Direkte avbildning (3/3)

- Hvordan sjekker man om noe ligger i hurtigb.?
  - Se på LINE-feltet i adressen
  - Slå opp på hurtigbufferlinja med det nummeret
  - Matcher TAG-feltet i adressen og hurtigbufferlinja?
- Hvor i hurtigbuffer kan nye linjer plasseres?
  - Gitt av LINE-feltet i adressen
- Hva skal kastes ut hvis hurtigbuffer er fullt?
  - Det som lå på hurtigbufferlinja fra før

Hurtigbuffer skriveoperasjonar

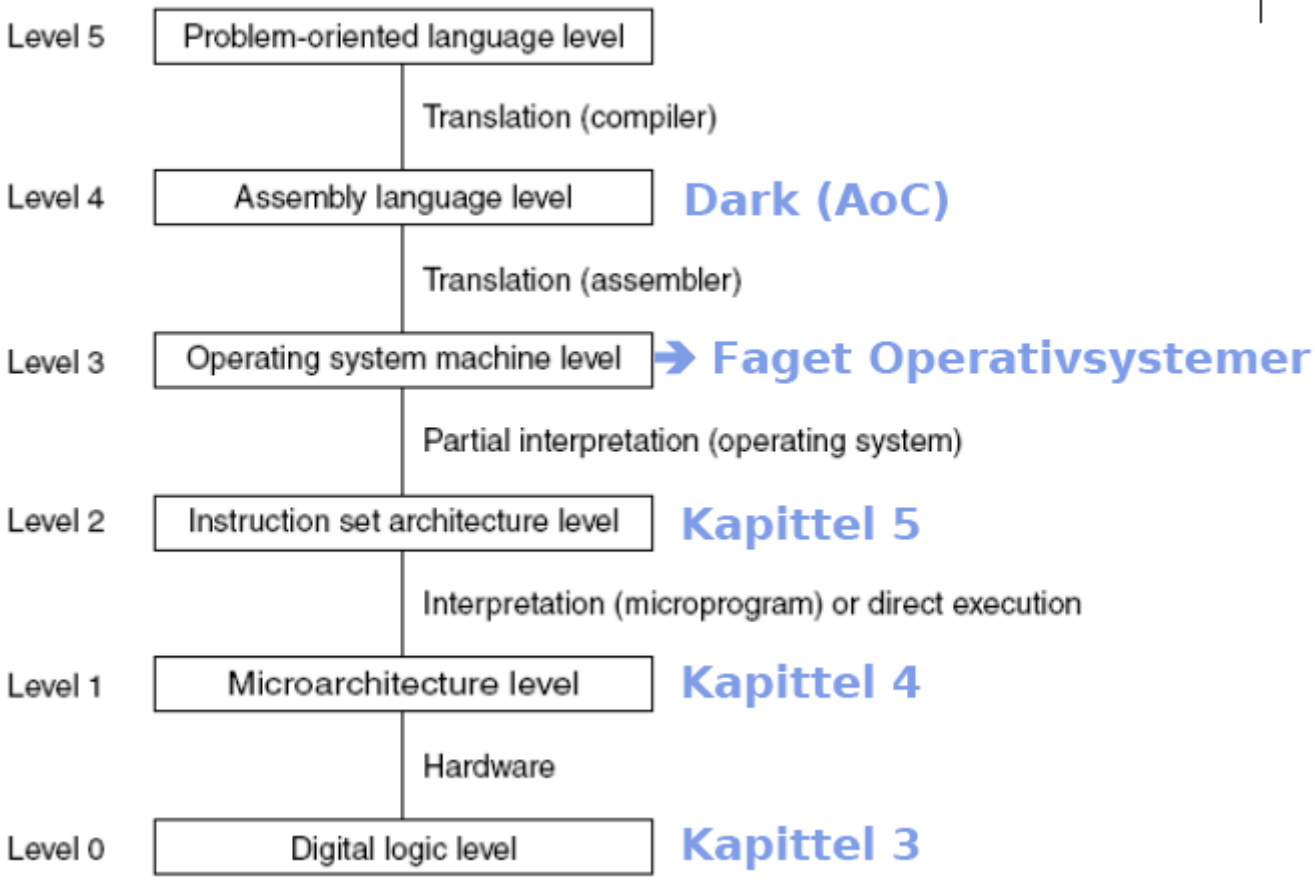
Prosessor ønsker å skrive til hovedlager

- Data **finnes** i hurtigbuffer:
  - Write through: Oppdater både hurtigbuffer og hovedlager
  - Write back: Oppdater bare hurtigbuffer
    - Oppdater hovedlager senere
- Data **finnes ikke** i hurtigbuffer:
  - Kan omgå hurtigbuffer helt
  - Write allocation: Hent data inn til hurtigbuffer først



ISA Instruction Set Architecture (5)

ISA (5)

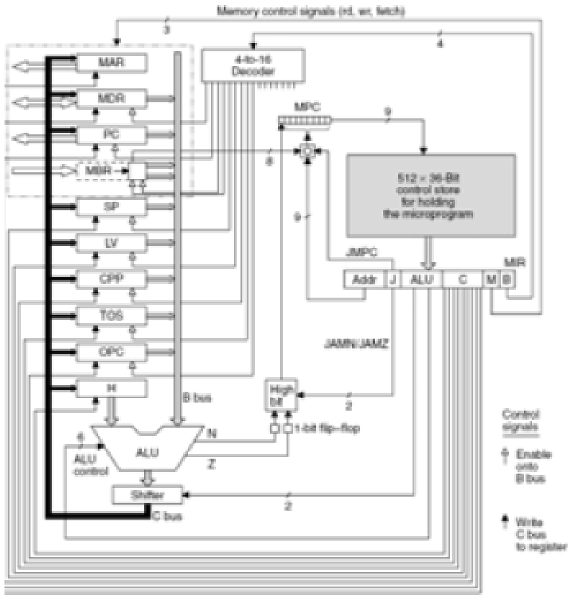

Level 5
Problem-oriented language level
Translation (compiler)
Level 4
Assembly language level
Dark (AoC)
Translation (assembler)
Level 3
Operating system machine level
Faget Operativsystemer
Partial interpretation (operating system)
Level 2
Instruction set architecture level
Kapittel 5
Interpretation (microprogram) or direct execution
Level 1
Microarchitecture level
Kapittel 4
Hardware
Level 0
Digital logic level
Kapittel 3

ISA vs. mikroarkitektur

ILOAD j
ILOAD k
IADD
ISTORE i
ILOAD i
BIPUSH 3
…

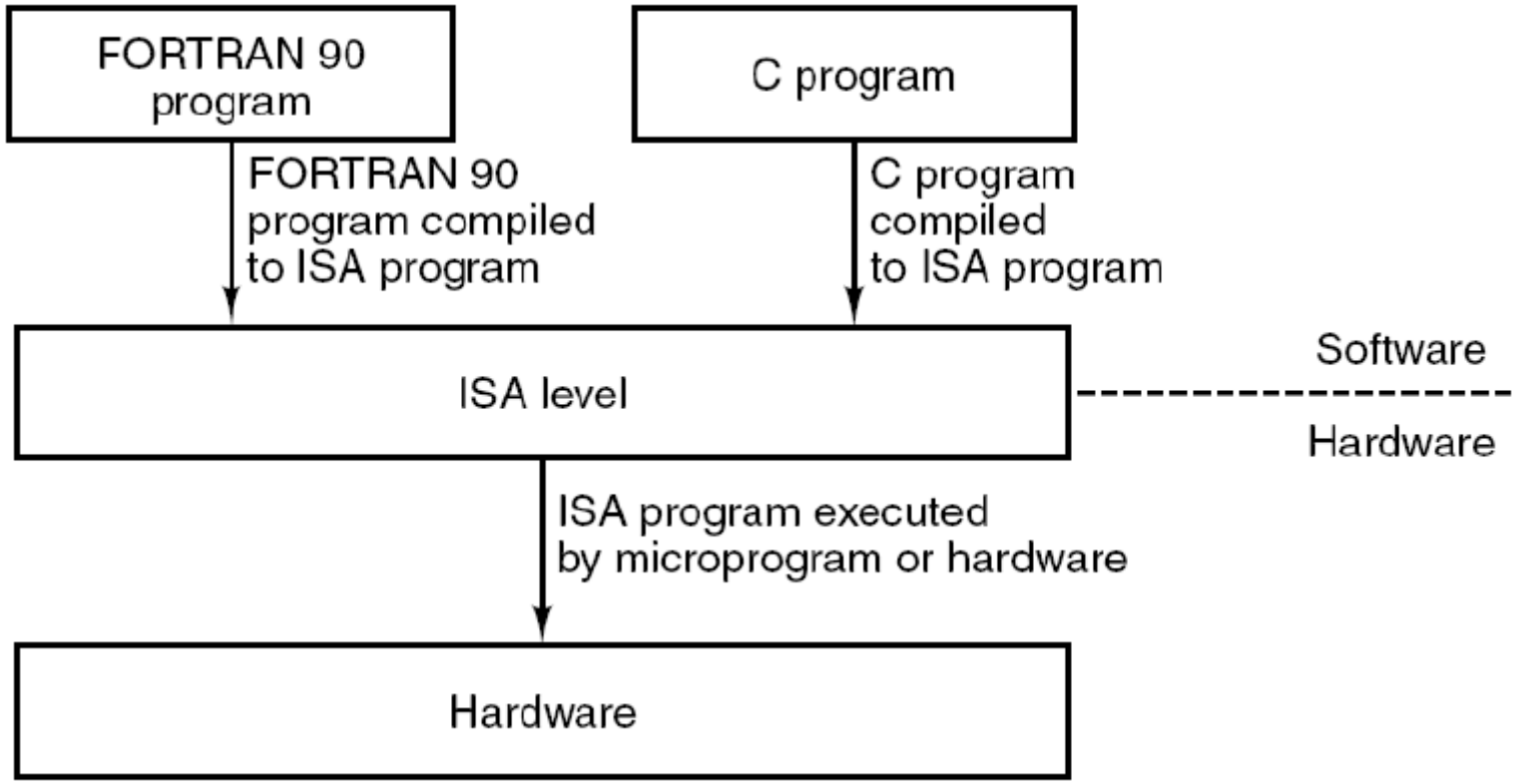
ISA-instr.

NEXT_ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F_0	F_1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	PC	MDR	MAR	WRITE	READ	FETCH	B bus
NEXT_ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F_0	F_1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	PC	MDR	MAR	WRITE	READ	FETCH	B bus
NEXT_ADDRESS	JMPC	JAMN	JAMZ	SLL8	SRA1	F_0	F_1	ENA	ENB	INVA	INC	H	OPC	TOS	CPP	LV	SP	PC	MDR	MAR	WRITE	READ	FETCH	B bus



Mikroprogram: Sekvens av µInstr.

ISA (5)


FORTRAN 90 program
C program
FORTRAN 90 program compiled to ISA program
C program compiled to ISA program
ISA level
Software
Hardware
ISA program executed by microprogram or hardware
Hardware

ISA (5)

- Opprineleg det einaste nivået
  - Ikkje skilje (microArc, ISA), huks ledningar og brytarar som programmering. Programmering direkte på samankopling
  - Ofte kalla ”architectur”
    - Funksjonelle einingar
    - Samankopling
    - Gir kva instruksjonar som er mest anvendelege
  - Grense mellom maskinvare og programvare
    - Maskinvare dårleg egna til å utføre C/C++/JAVA code
    - Maskina kan utføre maskinkode (ISA)
    - Kompilerar til ISA-code
      - Kan optimalisera til mikroarkitektur (486 med utan flyttal eining)
  - Kan definer ISA lage forskjelige mikroarkitekturar
    - Bakoverkompatibel
    - Ved ny mikroarkitektur
      - Nye instruksjonar
      - Auka yting (pipline, superskalaritet…)

ISA (5)

- Kva er ein bra ISA
  - Godt design gir auka yting
  - To målgrupper for ISA
    - Maskinvaredesignar
      - Implementere ein ISA effektivt (må vere effektiv for maskinvare)
    - Programvaredesignar
      - Enkelt å generere ISA-kode (kompilator)
        - Støtte datastrukturar
        - Støtte funksjonar (vanlege) effektivt
        - Må kunne kompilere program til ISA-kode som er effektiv (prøv å skrive C-prog for berekningar på IJVM, **NOT**)