ISA Instruction Set Architecture (5)

Instruksjonar

- Hvilke instruksjoner som finnes og hvordan de fungerer er helt sentralt i ISA-nivået
- Viktigste typer:
  - Flytting av data
  - Aritmetiske operasjoner
  - Sammenligninger og hoppinstruksjoner
  - Prosedyrekall
  - Løkker
  - I/O

NTNU
Innovation and Creativity

Instruksjonstypar I/O avbrudd

- Programstyrt I/O
  - Prosessor sjekker om data er klart
- Avbruddsinitiert I/O
  - I/O-enhet sier ifra når data er klart
  - Unngår å kaste bort prosessortid
- Adresse til avbruddsrutine
  - Ikke-vektorisert: Fast, en for alle avbrudd
  - Vektorisert: Avbruddskilde oppgir adresse
- Mer om avbrudd senere

Instruksjonstypar DMA

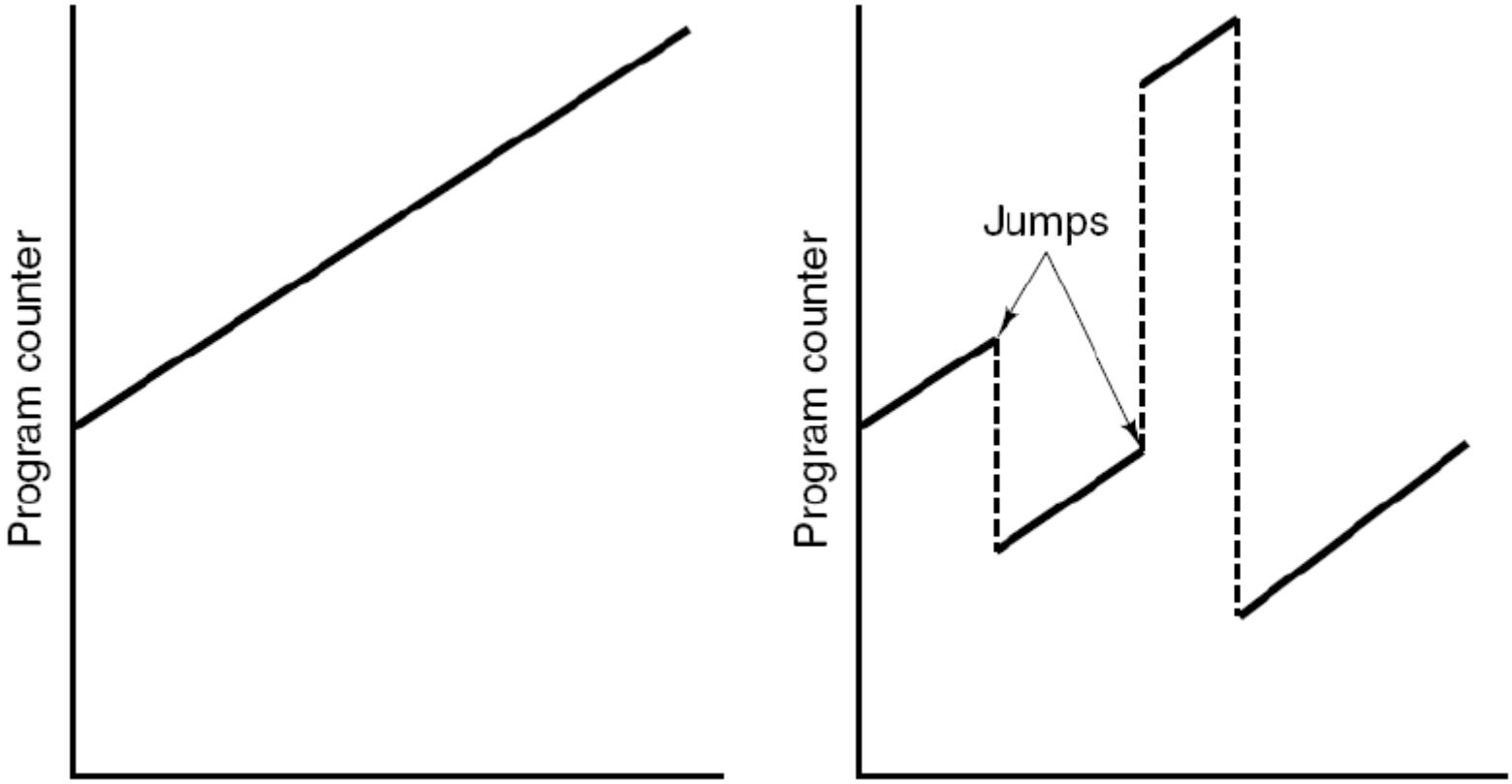
- Mye dataoverføring gir prosessor lite tid til annet
- Kan ikke andre ta jobben?
- Direct Memory Access
  - Overføring direkte til/fra hovedlager
  - Prosessor tar seg kun av oppstart av overføring
    - Enhet, startadresse, datalengde
- Prosessor kan dermed gjøre annet imens
  - Men: Ikke aksessere hovedlager!
  - DMA høyere prioritet enn CPU, kan ofte ikke vente
  - DMA ”stjeler” buss-sykler fra CPU – ”cycle stealing”

Flytkontrol (flow control)

- Normalt: Sekvensiell utføring
- Flytkontroll: Dynamisk endring av instruksjonsrekkefølge under utføring
- Typer
  - (Betinget) Hopp
  - Prosedyrekall (Java: Metodekall)
  - Ko-rutiner
  - Trap / Avbrudd (”Interrupt”)

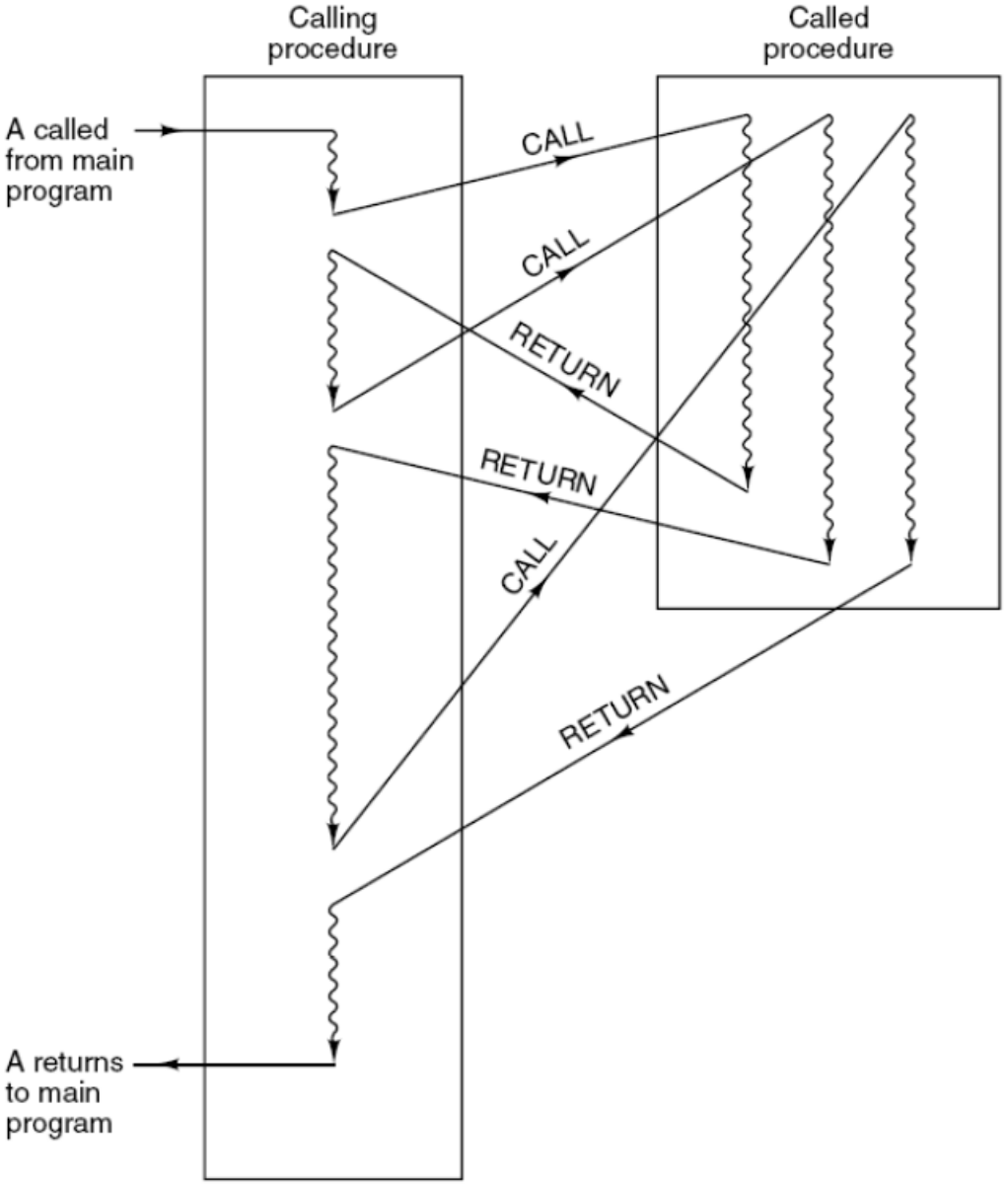
NTNU
Innovation and Creativity

Flytkontrol (flow control)

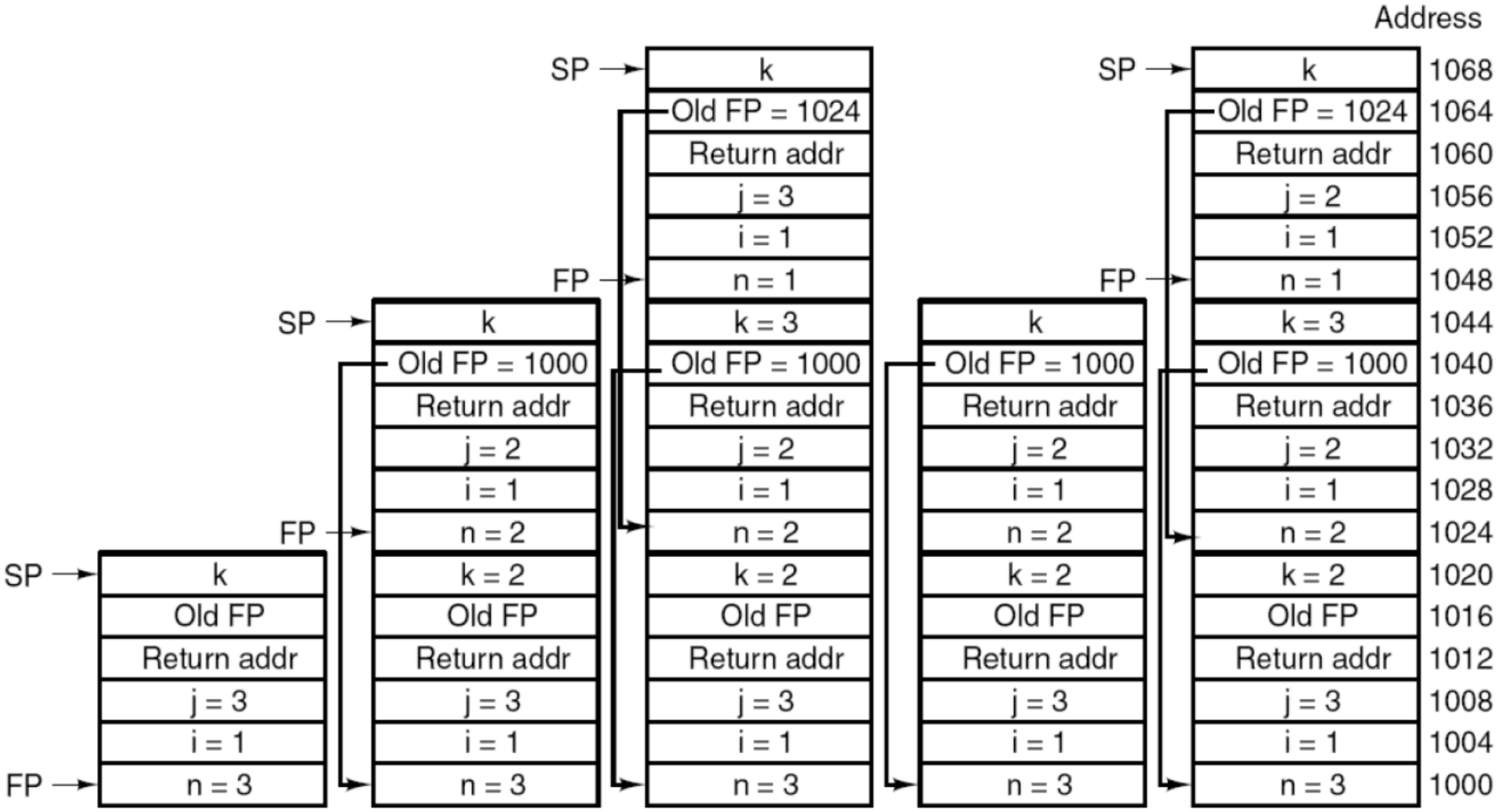


Prosedyre

- Viktig del av strukturert programmering
- Gir mulighet for abstraksjon
  - Kan se en prosedyre som en instruksjon
- Økt lesbarhet
- På lavt nivå er det egentlig hopp likevel



Prosedyre (stack)



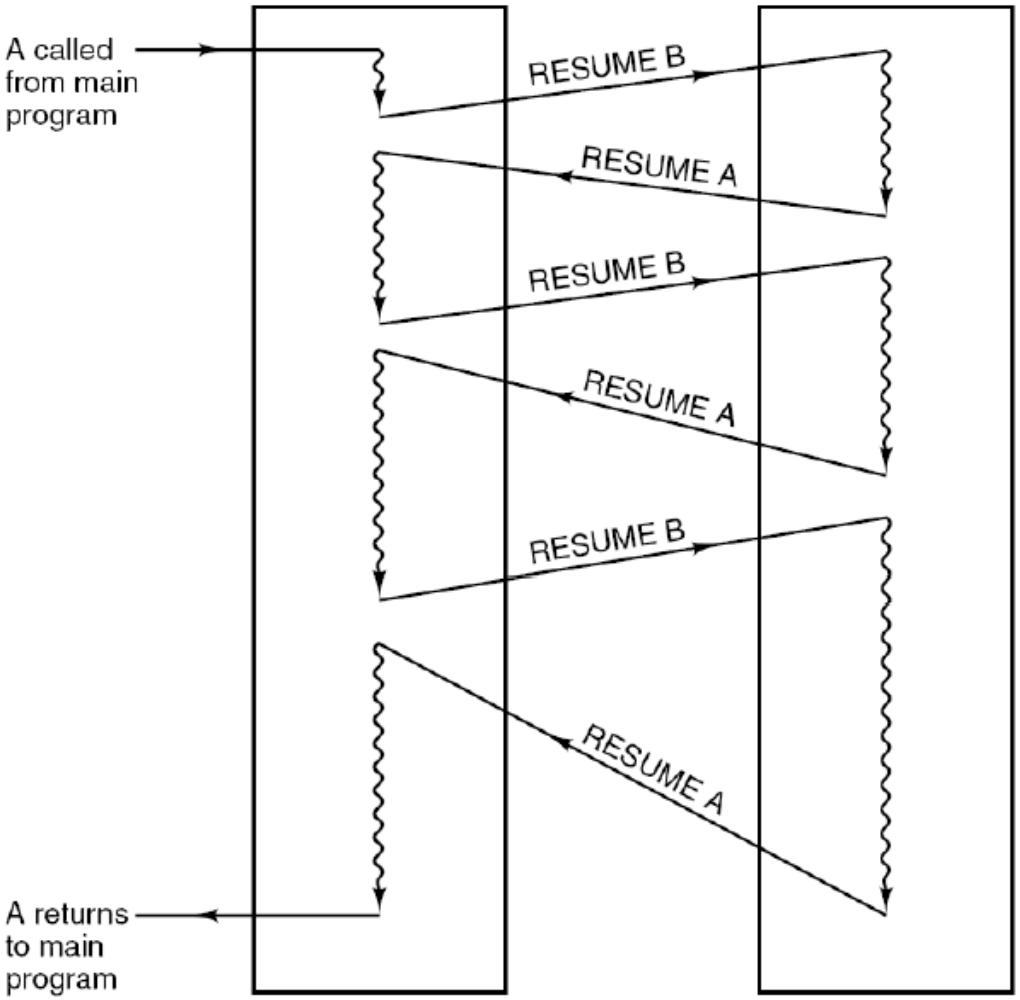
Prosedyre oppsummering

- Bruk av stakk sentralt
- Overhead både ved
  - Start av prosedyre – *prolog*
  - Slutt av prosedyre – *epilog*
- Kan ha spesialinstruksjoner i ISA-nivået for at dette skal gå fort
  - Eksempel: Pentium 4 – ENTER / LEAVE

NTNU
Innovation and Creativity

Ko-rutiner

- Når en prosedyre blir kalt, utføres den fra starten av
- Ko-rutiner: Fortsetter der utførelsen stoppet sist
  - Kan ikke bruke CALL/RETURN
  - Sjelden maskinvarestøttet
- Kan brukes til å simulere flere prosessorer – flere programmer utføres ”samtidig”



NTNU
Innovation and Creativity

Trap

- Automatisk prosedyrekall dersom ”noe” skjer
  - Tenk try-catch i Java
- Mulige hendelser
  - Overflyt (heltall/flyttall), underflyt (flyttall)
  - Divisjon med 0
  - Ugyldig opkode
  - Overflyt på stakk
- Sjekk kan gjøres…
  - på mikroarkitektur-nivå
    (mikroprogram el. maskinvare)
    Programmerer oppgir adresse for hånteringsrutine
  - på ISA-nivå (programmerer tester selv)

Avbrudd

- Automatisk prosedyrekall dersom ”noe” skjer
- Men: Hendelse utenfor program
  - Harddisk klar til å motta data, tast trykket osv.
  - Trap: Hendelse innenfor program – divisjon på 0
  - Kan derfor oppstå når som helst
- Når avbrudd skjer
  - Utfør avbruddsrutine
  - Returner tilbake til program
- Transparent – når avbruddet er behandlet, er alt slik det var (samme tilstand)

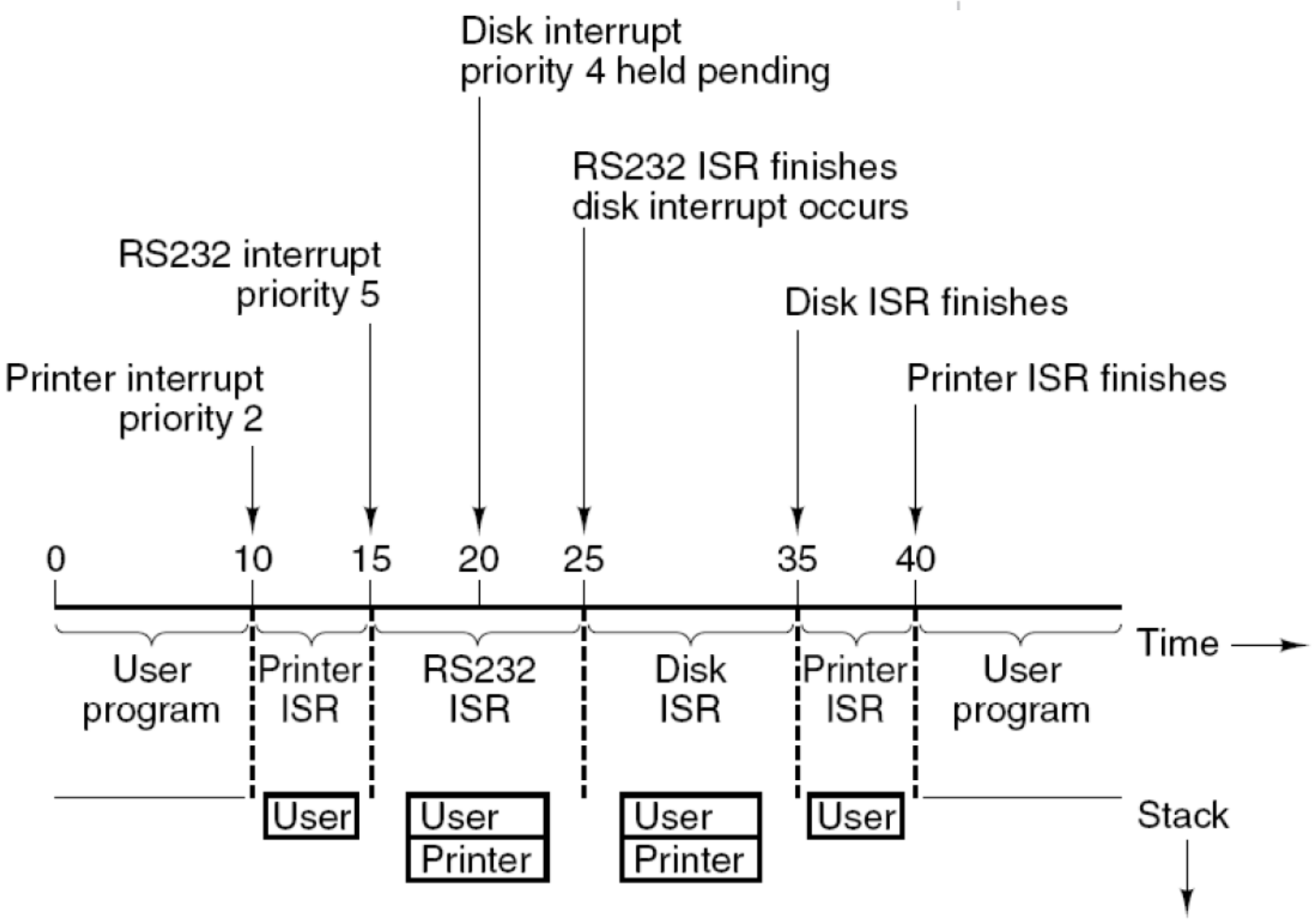
Avbrudd (utføring)

1. Enhet aktiverer avbruddslinje
2. CPU godkjenner avbrudd
3. Enhet oppgir avbruddsvektor
4. CPU lagrer programinformasjon (PC, PSW)
5. CPU bruker avbruddsvektor til å finne avbruddsrutine
    1. Avbruddsrutine lagrer registre
    2. Avbruddsrutine finner info om avbrudd
    3. Avbrudd håndteres
    4. Avbruddsrutine setter tilbake registerverdier
6. PC og PSW tilbakestilles

Avbrudd prioritering

- Hva hvis to avbrudd skjer samtidig?
- Hva hvis avbrudd kommer i avbruddsrutine?
- Trenger system for prioritering
  - Høy prioritet behandles før lav
  - Høy prioritet kan avbryte avbruddsrutine til lav
- Tre mulige løsninger:
  - Programvarebasert
    - Sjekker alle mulige kilder: Var det deg?
    - Mange avbrudd → Tar for lang tid
  - Maskinvarebasert, seriell
  - Maskinvarebasert, parallell

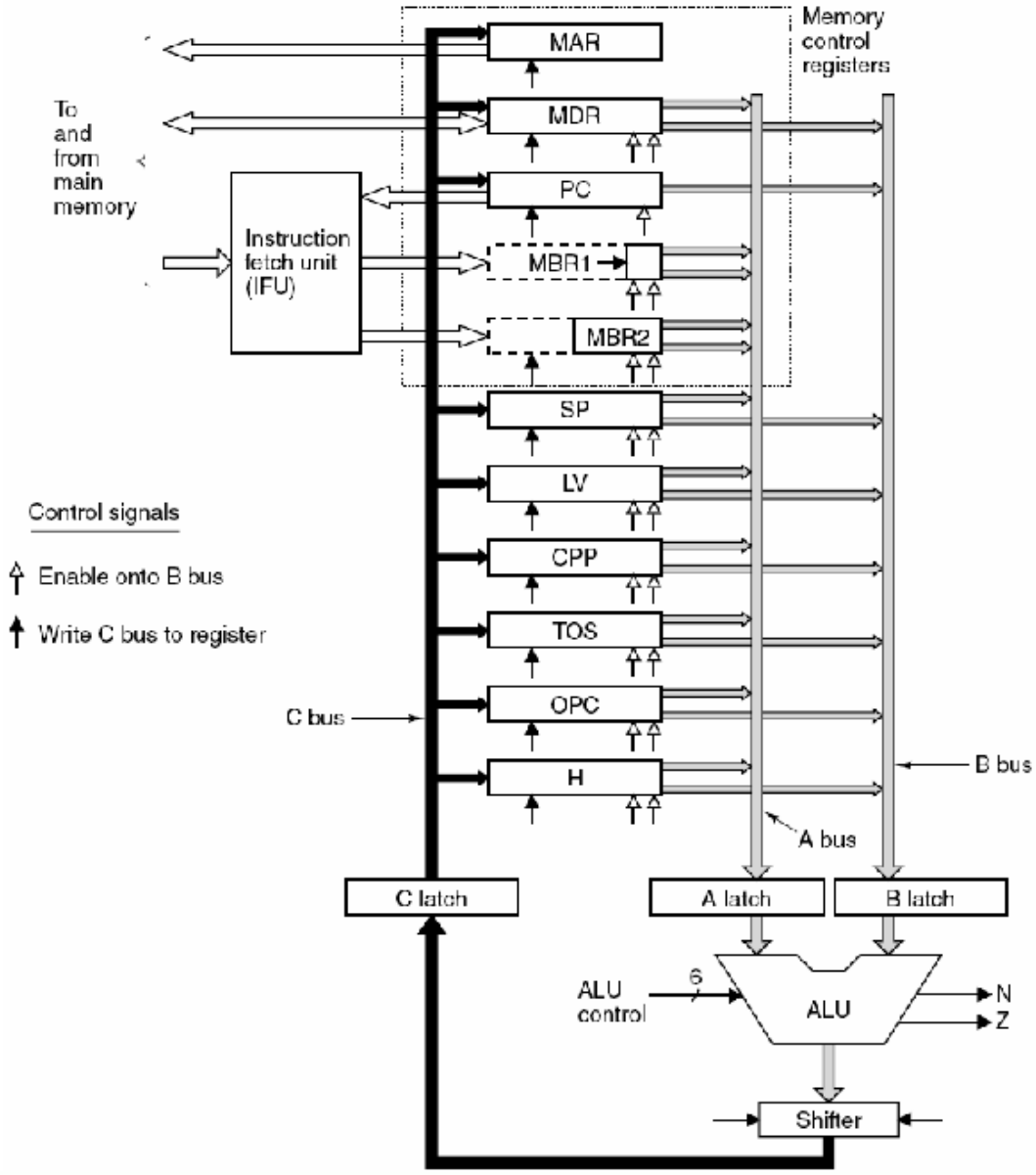
Avbrudd prioritering



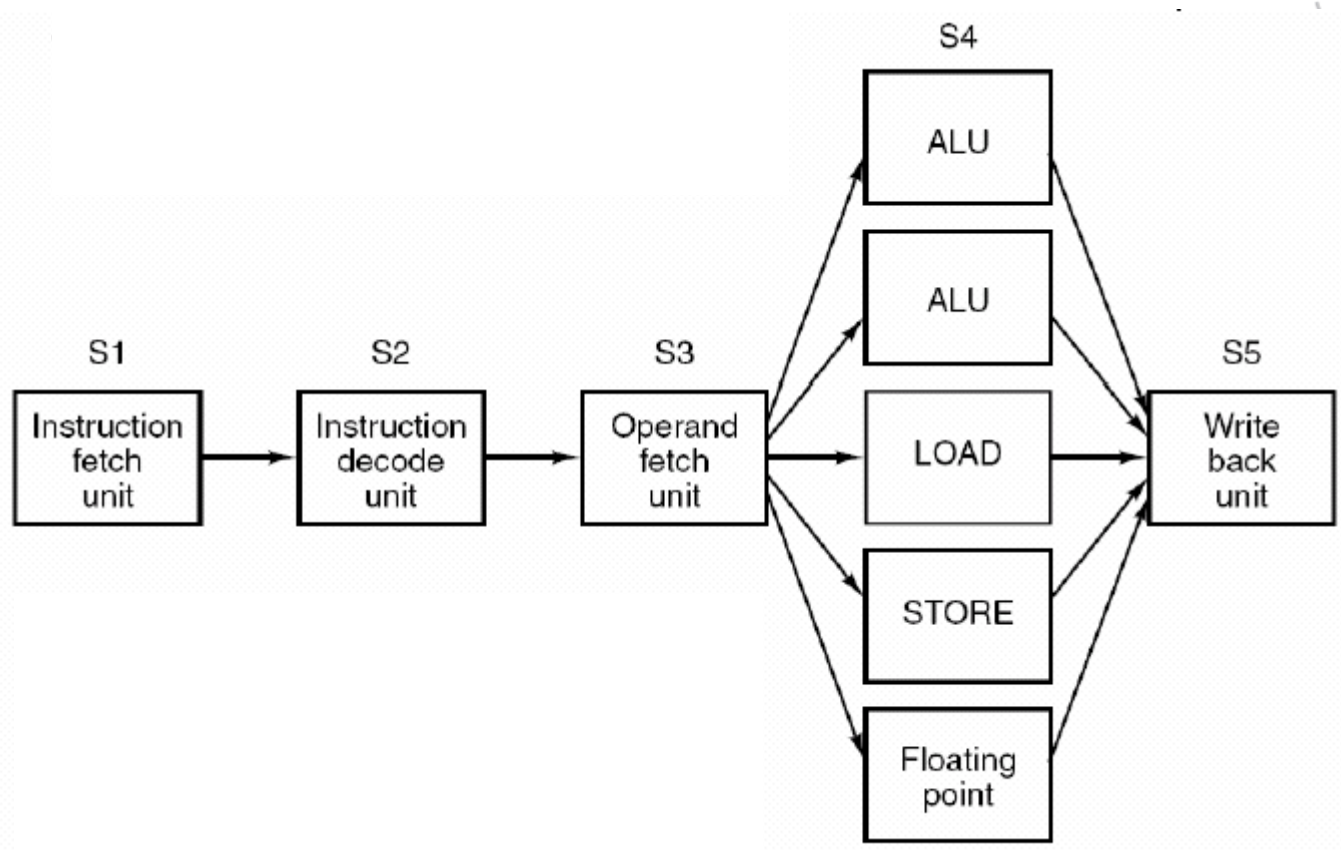
Scordboarding

- Scoreboarding er en metode for å finne avhengigheter mellom instruksjoner
- CPU'en har en tabell over hvilke registere som er I bruk.
- Out-of-order issue: Instruksjoner kan startes I vilkårlig rekkefølge. (Motsatt: in-order)
- Out-of-order commit: Instruksjoner kan avsluttes I vilkårlig rekkefølge (Motsatt: in-order)

Scordboarding



Scordboarding (superskalaritet)



Scordboarding

- En instruksjon kan bare starte opp dersom:
- Ingen av registrene den leser fra blir skrevet til av en annen instruksjon. (RAW)
- Ingen leser fra registeret den skriver til. (WAR)
- Ingen skriver til registeret den skriver til. (WAW)
- Funksjonelle enheter er ledig.

					Registers Read								Registers Written							
Cy	#	Decoded	Iss	Ret	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
1	1	Add R1, R2, R3	1				1	1						1						
	2	Add R4, R2, R3	2				2	2						1			1			
2	3	Sub R5, R1, R4	--				2	2						1			1			
							2	2						1			1			
3				1			1	1									1			
				2																
			3			1			1									1		
	4	Mul R4, R2, R3	-			1			1									1		
4						1			1									1		
						1			1									1		
5				3																
			4				1	1									1			
	5	Add R6, R2, R1	5			1	2	1									1		1	
6	6	Sub R0, R5, R3	6			1	2	2		1			1				1		1	
	7	Div R7, R1, R6	-			1	2	2		1			1				1		1	
7				4		1	1	1		1			1						1	
				5				1		1			1							
			7			1		1		1	1		1							1
	8	Add R6, R1, R6	8			2		1		1	2		1						1	1
8				6		2					2								1	1
						2					2								1	1
9				7		1					1								1	
				8																

NTNU
Innovation and Creativity

Scordboarding

- Først setter man instruksjonsnummeret i kolonnen for "issued"
- For alle registere som blir lest, øk tallet i tilsvarende kolonne.
- For registeret som blir skrevet, sett rubrikken til 1. (Står det et ett-tall der fra før, så har du gjort noe feil.
- Når instruksjonen er ferdig, set instruksjonsnummeret i "retired" kolonnen, og gjør skritt 2 og 3 motsatt.

					Registers Read								Registers Written							
Cy	#	Decoded	Iss	Ret	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
1	1	Add R1, R2, R3	1				1	1						1						
	2	Add R4, R2, R3	2				2	2						1			1			
2	3	Sub R5, R1, R4	--				2	2						1			1			
							2	2						1			1			
3				1			1	1									1			
				2																
			3			1			1									1		
	4	Mul R4, R2, R3	-			1			1									1		
4						1			1									1		
						1			1									1		
5				3																
			4				1	1									1			
	5	Add R6, R2, R1	5			1	2	1									1		1	
6	6	Sub R0, R5, R3	6			1	2	2		1			1				1		1	
	7	Div R7, R1, R6	-			1	2	2		1			1				1		1	
7				4		1	1	1		1			1						1	
				5				1		1			1							
			7			1		1		1	1		1							1
	8	Add R6, R1, R6	8			2		1		1	2		1						1	1
8				6		2					2								1	1
						2					2								1	1
9				7		1					1								1	
				8																

NTNU
Innovation and Creativity

Scordboarding

- Først setter man instruksjonsnummeret i kolonnen for "issued"
- For alle registere som blir lest, øk tallet i tilsvarende kolonne.
- For registeret som blir skrevet, sett rubrikken til 1. (Står det et ett-tall der fra før, så har du gjort noe feil.
- Når instruksjonen er ferdig, set instruksjonsnummeret i "retired" kolonnen, og gjør skritt 2 og 3 motsatt.

Scordboarding

Add R1, R2, R3

Add R4, R2, R3

Sub R5, R1, R4 (RAW – I1, RAW - I2)

Mul R5, R2, R3 (WAW - I3)

Add R6, R2, R1 (RAW – I1)

Sub R0, R5, R3 (RAW – I4)

Div R7, R1, R6 (RAW – I5, RAW - I1)

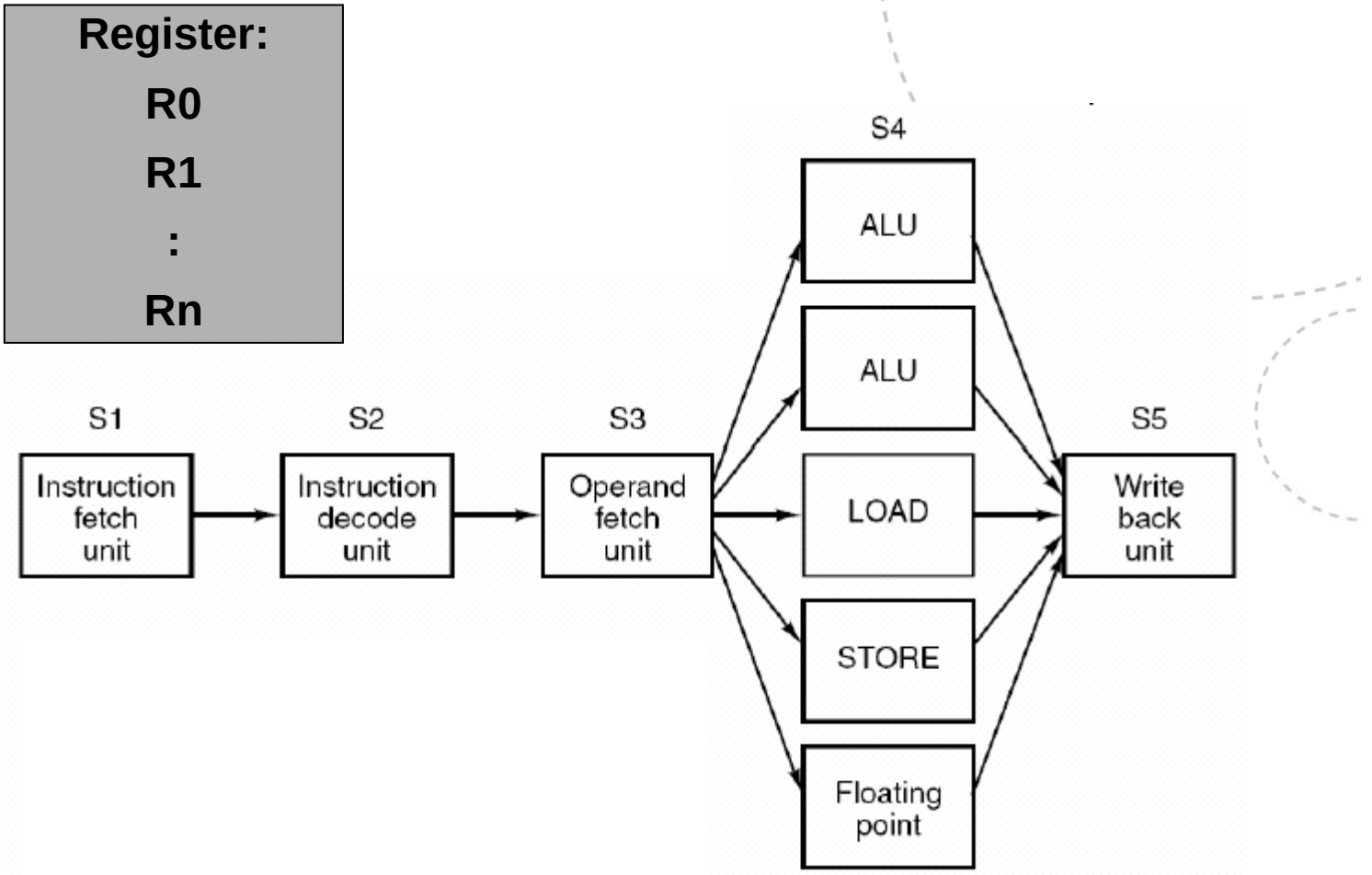
Add R6, R1, R6 (WAR - I7, RAW – I1, RAW - I

Scordboarding

1. Add R1, R2, R3
2. Add R4, R2, R3
3. Sub R5, R1, R4 (RAW – I1, RAW - I2)
4. Mul R5, R2, R3 (WAW - I3)
5. Add R6, R2, R1 (RAW – I1)
6. Sub R0, R5, R3 (RAW – I4)
7. Div R7, R1, R6 (RAW – I5, RAW - I1)
8. Add R6, R1, R6 (WAR - I7, RAW – I1, RAW - I5)

					Registers Read								Registers Written							
Cy	#	Decoded	Iss	Ret	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
1	1	Add R1, R2, R3	1				1	1						1						
	2	Add R4, R2, R3	2				2	2						1			1			
2	3	Sub R5, R1, R4	--				2	2						1			1			
							2	2						1			1			
3				1			1	1									1			
				2																
			3			1			1									1		
	4	Mul R4, R2, R3	-			1			1									1		
4						1			1									1		
						1			1									1		
5				3																
			4				1	1									1			
	5	Add R6, R2, R1	5			1	2	1									1		1	
6	6	Sub R0, R5, R3	6			1	2	2		1			1				1		1	
	7	Div R7, R1, R6	-			1	2	2		1			1				1		1	
7				4		1	1	1		1			1						1	
				5				1		1			1							
			7			1		1		1	1		1							1
	8	Add R6, R1, R6	8			2		1		1	2		1						1	1
8				6		2					2								1	1
						2					2								1	1
9				7		1					1								1	
				8																

NTNU
Innovation and Creativity


Register:
R0
R1
:
Rn
S1
Instruction fetch unit
S2
Instruction decode unit
S3
Operand fetch unit
S4
ALU
ALU
LOAD
STORE
Floating point
S5
Write back unit