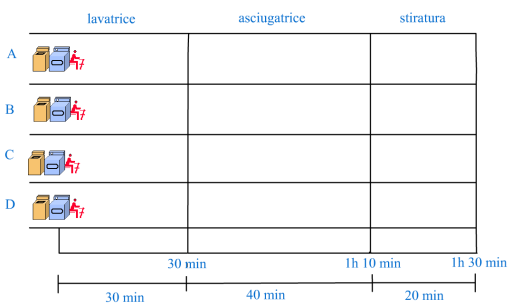
Organizzazione pipeline della CPU

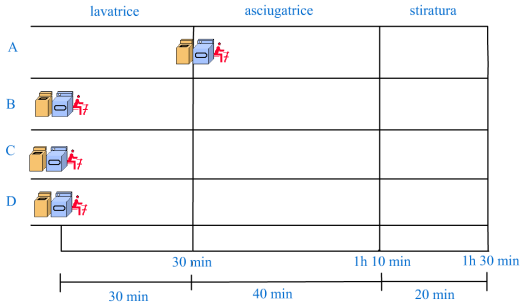
Riferimento:
D. A. Patterson, J.L. Hennessy, “Struttura e Progetto dei Calcolatori – L'interfaccia Hardware-Software” -II edizione Zanichelli editore, ISBN: 8808091457

Esecuzione sequenziale

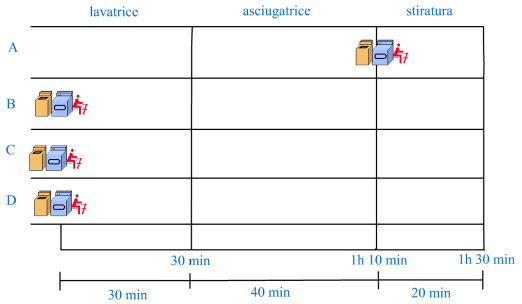
Prof. G. Ascia



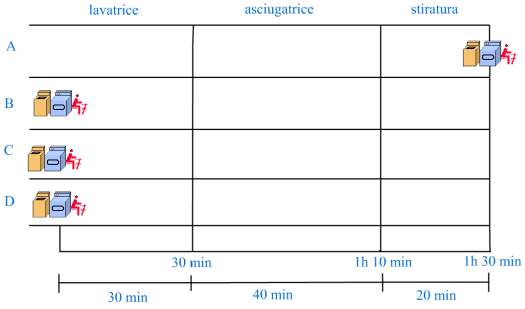
Tempo :



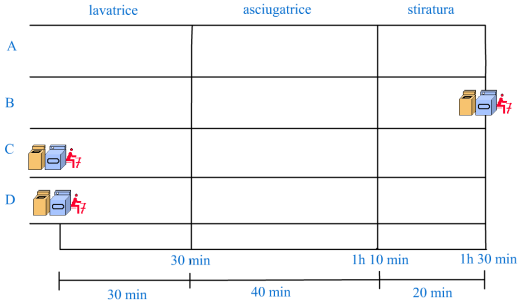
Tempo :30



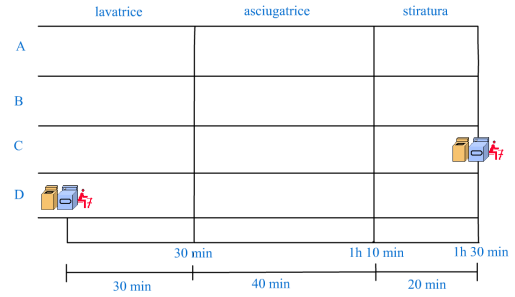
Tempo :30+40



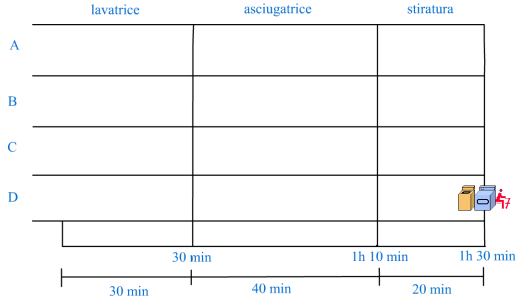
Tempo :30+40+20



Tempo :30+40+20+30+40+20



Tempo :30+40+20+30+40+20+30+40+20



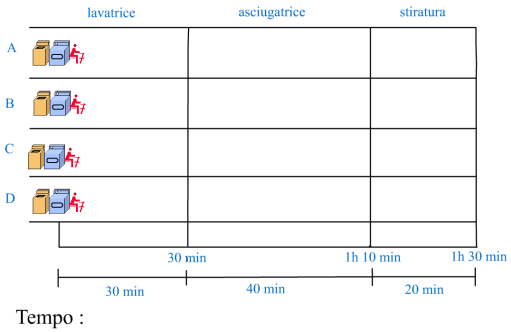
Tempo :30+40+20+30+40+20+30+40+20+30+40+20 = 6 h

01_Esecuzione_sequenziale.exe

Tempo = 4 * (30+40+20) = 6 h

Esecuzione pipeline

Prof. G. Ascia



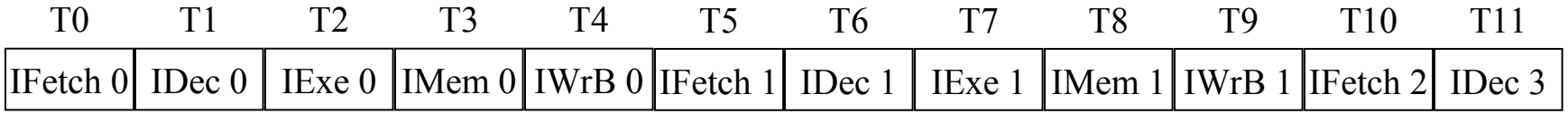
02_Esecuzione_pipeline.exe

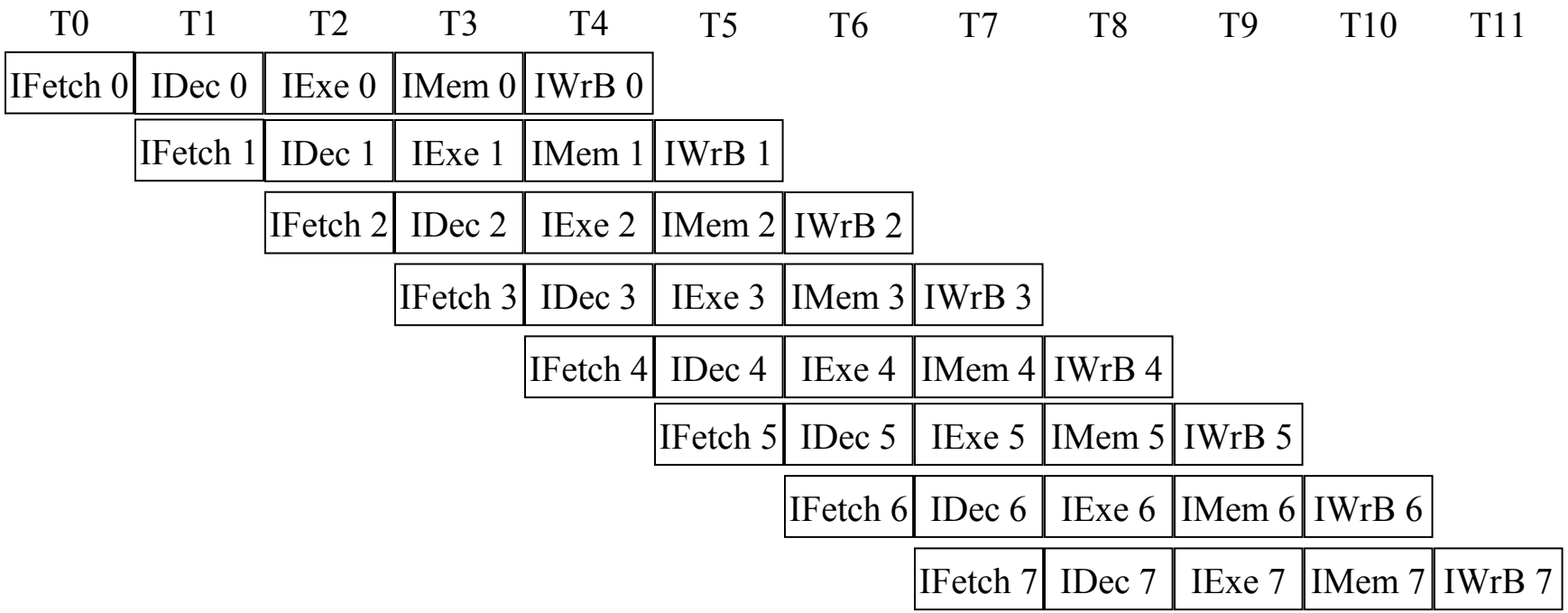
Pipeline

- **Il pipeline non riduce la latenza del singolo task, aumenta il throughput dell'intero workload**
- **Il pipeline rate è limitato dal più lento pipeline stage**
- **Più task operano simultaneamente**
- **Potenziale miglioramento = Numero di pipe stages**
- **Lunghezze non bilanciate dei pipeline stages riducono il miglioramento**

Esecuzione pipeline delle istruzioni

Prof. G. Ascia





Prestazione con pipeline

- Il tempo di exec della singola istruzione non diminuisce
- Il tempo medio di exec delle istruzioni si riduce di un fattore **N** (caso ideale)
- Il throughput migliora di un fattore **N**

- Una istruz. eseguita per CLK

→ $CPI_{Pipe} = 1$ $(CPI_{Sequenziale}=N)$

Prestazione con pipeline

Prof. G. Ascia

$$CPUtime_{Pipe} = IC \times CPI_{Pipe} \times T_{CK} = IC \times 1 \times T_{CK}$$

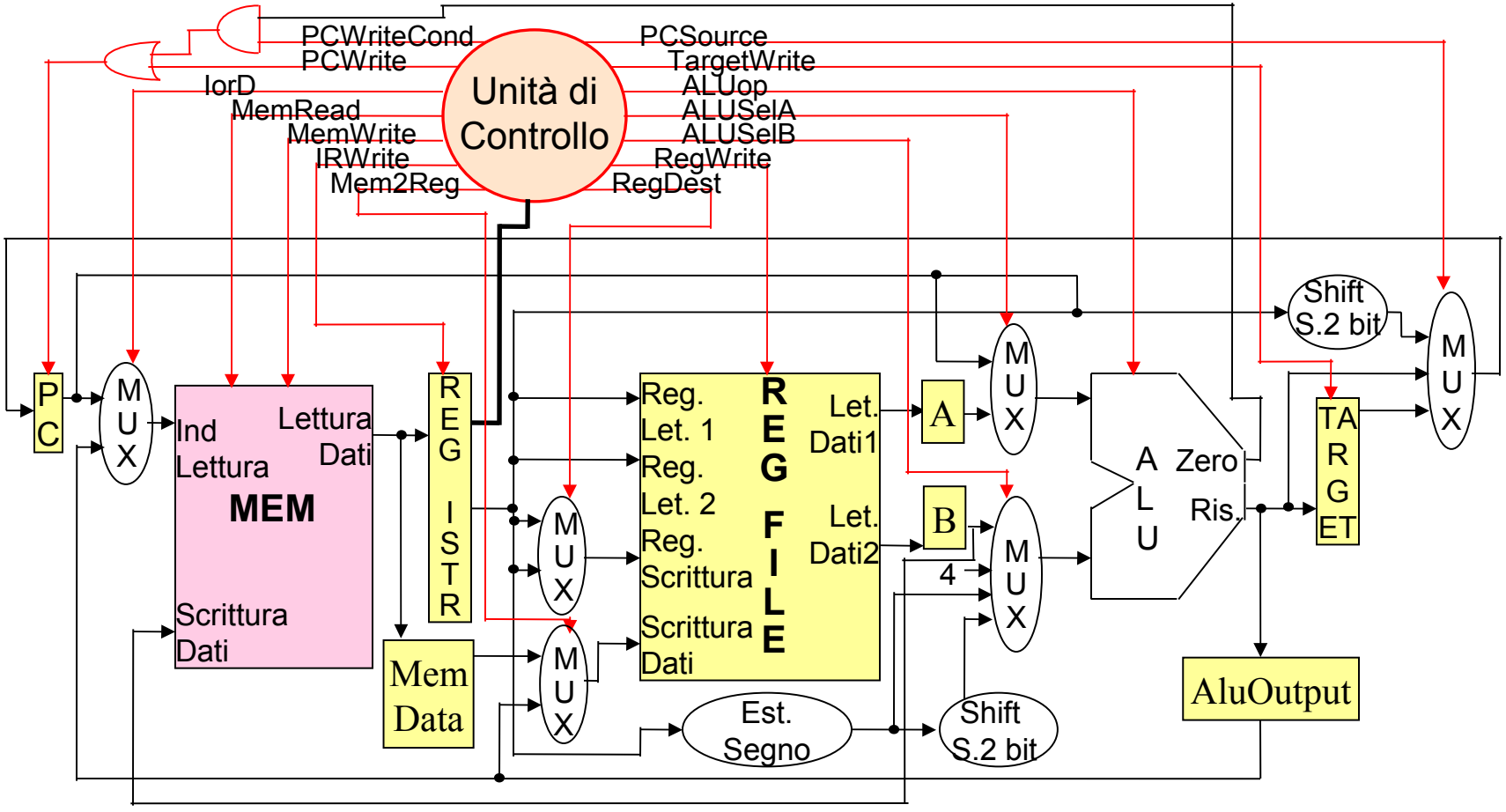
$$CPUtime_{Seq} = IC \times N \times T_{CK} = N \times CPUtime_{Pipe}$$

$$Speedup = \frac{CPU_{TIME\ Sequenziale}}{CPU_{TIME\ Pipeline}} = N$$

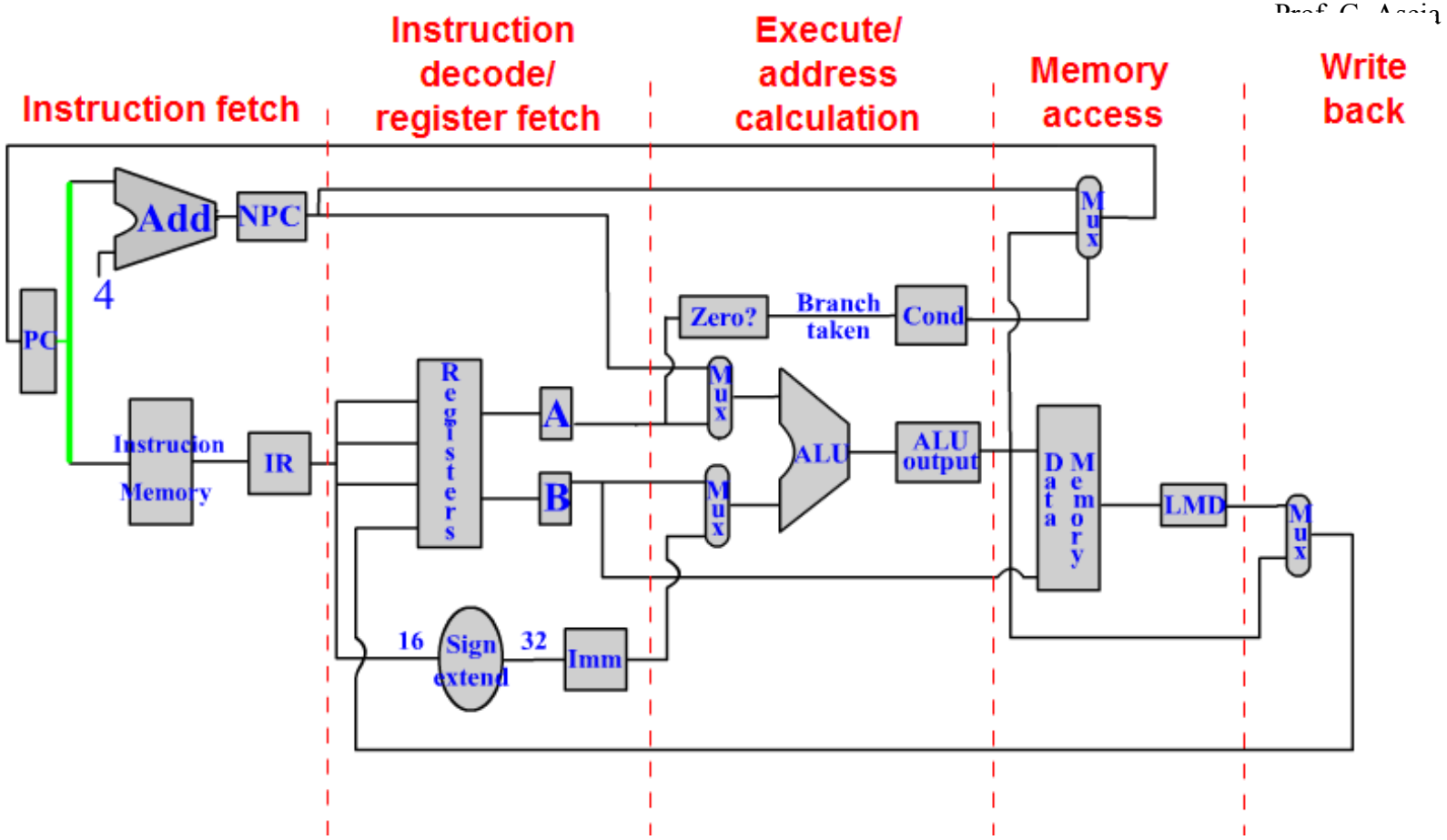
Ma questo è vero solo nel caso ideale!

DLX sequenziale multiciclo

Prof. G. Ascia



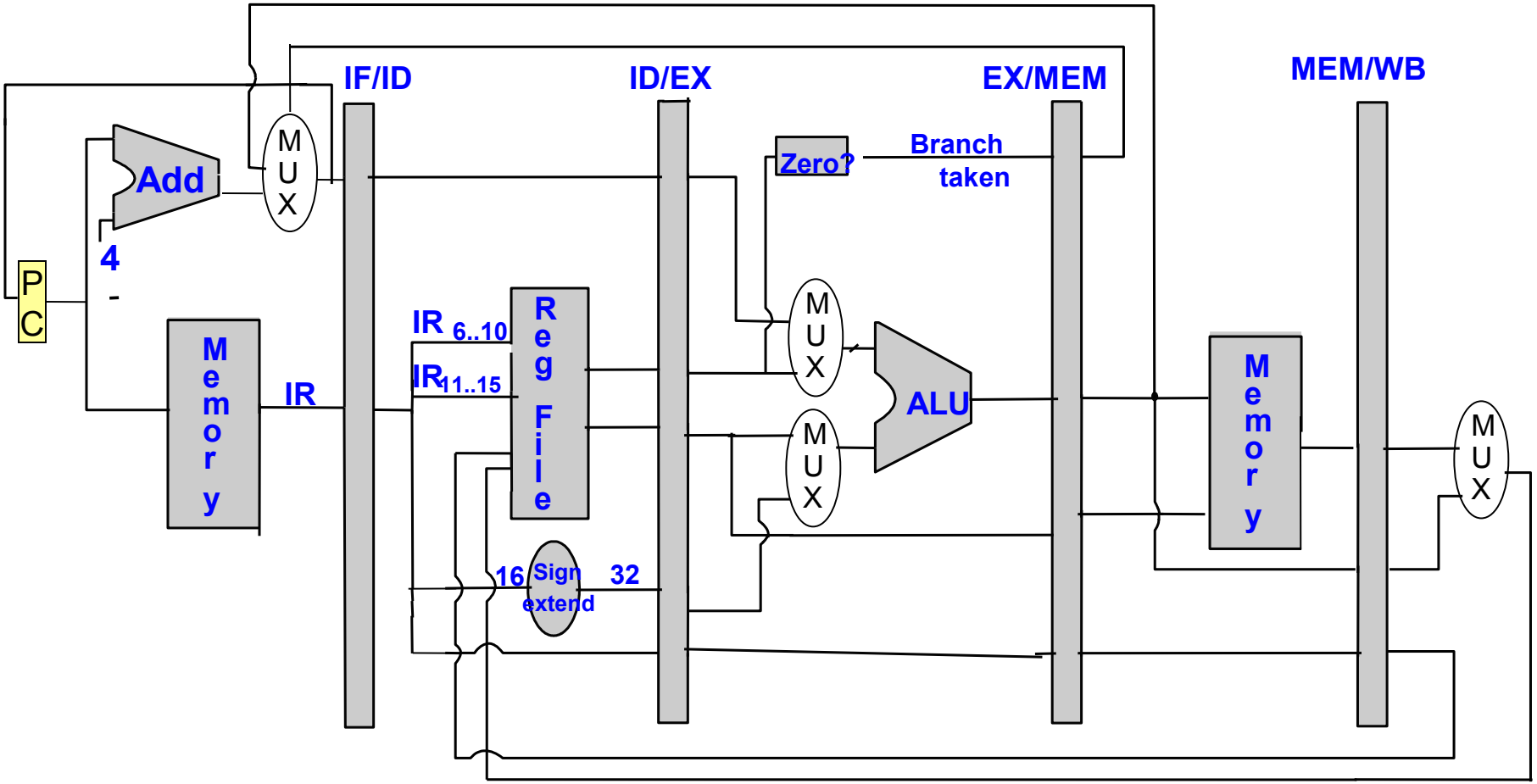
DLX sequenziale



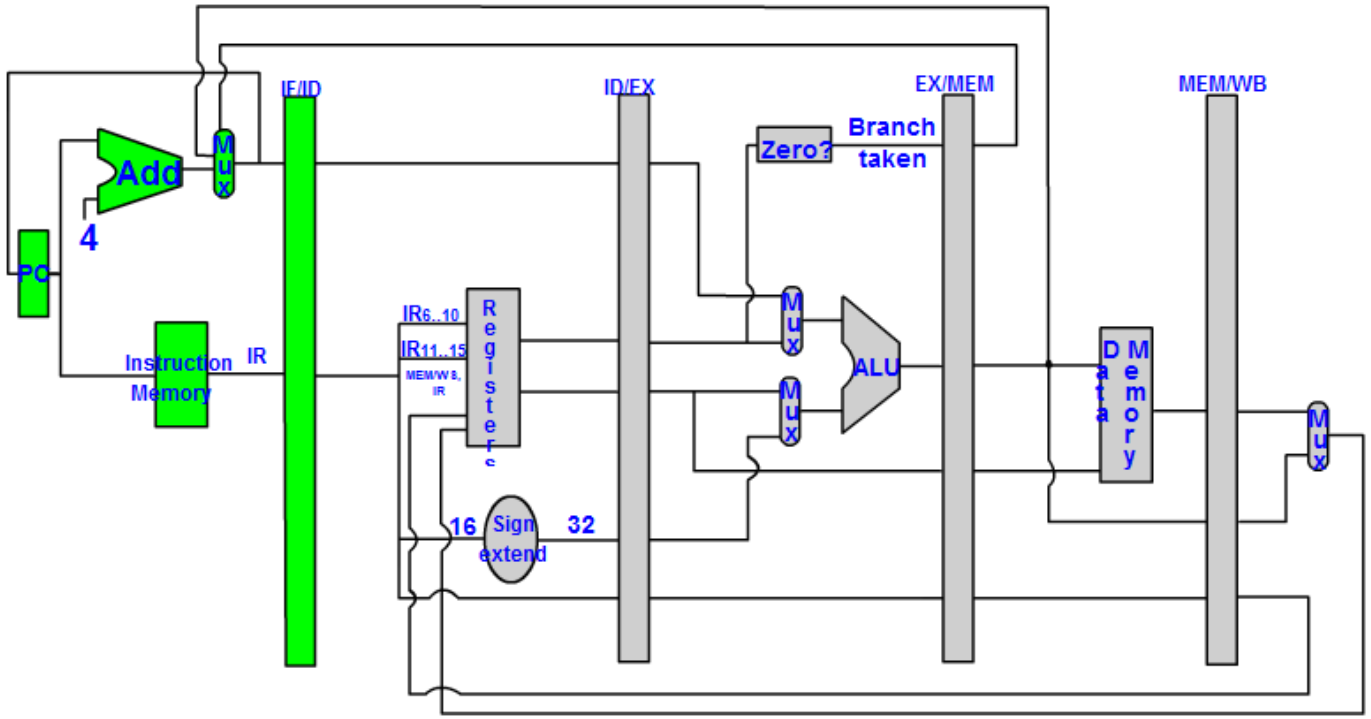
03_Datapath_DLX_sequenziale.exe

DLX pipeline

Prof. G. Ascia



Esecuzione pipeline DLX



04_Datapath_DLX_pipeline.exe

Pipeline DLX

Prof. G. Ascia

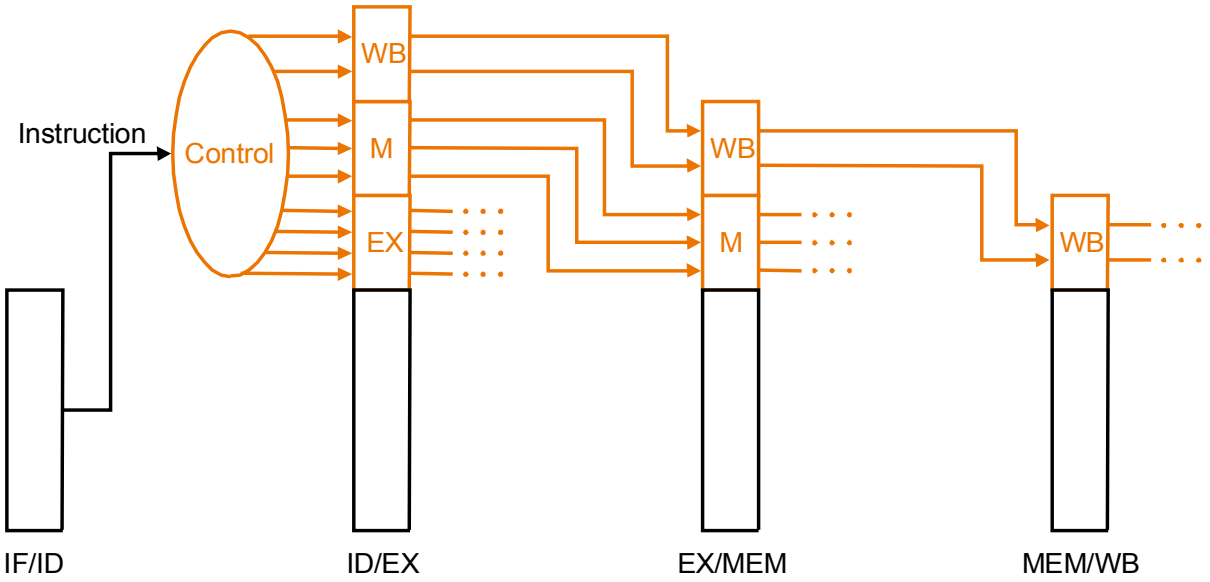
Stage	Any instruction
IF	IF/ID.IR ← Mem[PC]; IF/ID.NPC,PC ← (if EX/MEM.cond {EX/MEM.ALUOutput} else {PC+4});
ID	ID/EX.A ← Regs[IF/ID.IR$_{6..10}$]; ID/EX.B ← Regs[IF/ID.IR$_{11..15}$]; ID/EX.NPC ← IF/ID.NPC; IF/EX.IR ← IF/ID.IR; ID/EX.Imm ← (IF/ID.IR$_{16}$)^{16}##IF/ID.IR$_{16..31}$;

	ALU instruction	Load or store instruction	Branch instructiom
EX	EX/MEM.IR ← ID/EX.IR; EX/MEM.ALUOutput ← ID/EX.A *func* ID/EX.B; **or** EX/MEM.ALUOuput ← ID/EX.A *op* ID/EX.Imm; EX/MEM.cond ← 0;	EX/MEM.IR ← ID/EX.IR EX/MEM.ALUOutput ← ID/EX.A + ID/EX.Imm; EX/MEM.cond ← 0; EX/MEM.B ← ID/EX.B;	EX/MEM.ALUOutput ← ID/EX.NPC+ID/EX.Imm; EX/MEM.cond ← (ID/EX.A *op* 0);

Pipeline DLX

	ALU instruction	Load or store instruction	Branch instructiom
MEM	MEM/WB.IR ←EX/MEM.IR; MEM/WB.ALUOutput ← EX/MEM.ALUOutput;	MEM/WB.IR ← EX/MEM.IR; MEM/WB.LMD ← Mem[EX/MEM.ALUOutput]; **or** Mem[EX/MEM.ALUOutput] ← EX/MEM.B;	
WB	Regs[MEM/WB.IR$_{16..20}$] ← MEM/WB.ALUOutput; **or** Regs[MEM/WB.IR$_{11..15}$] ← MEM/WB.ALUOutput;	Regs[MEM/WB.IR$_{11..15}$] ← MEM/WB.LMD;	

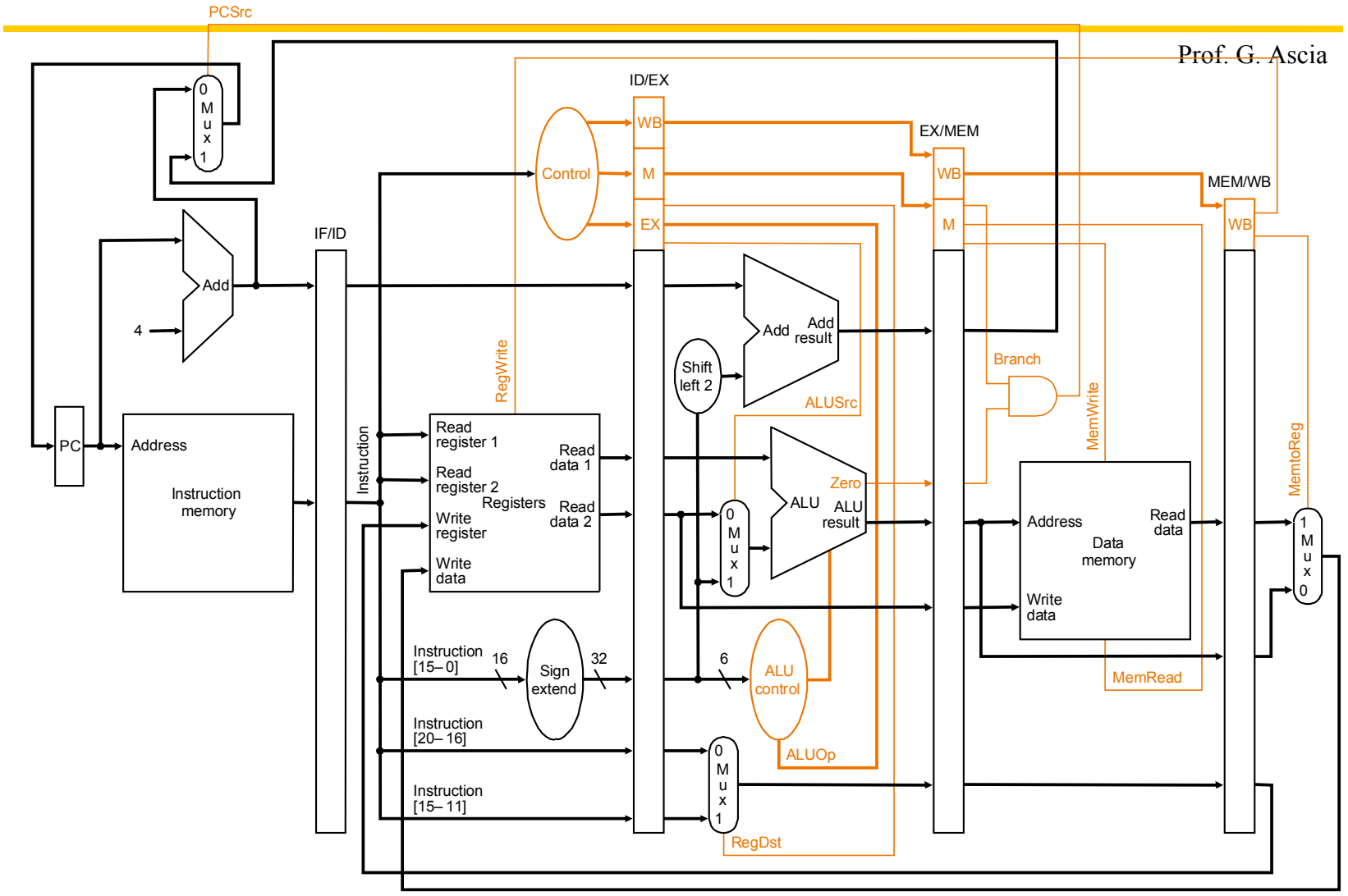
Unità di Controllo per il pipeline



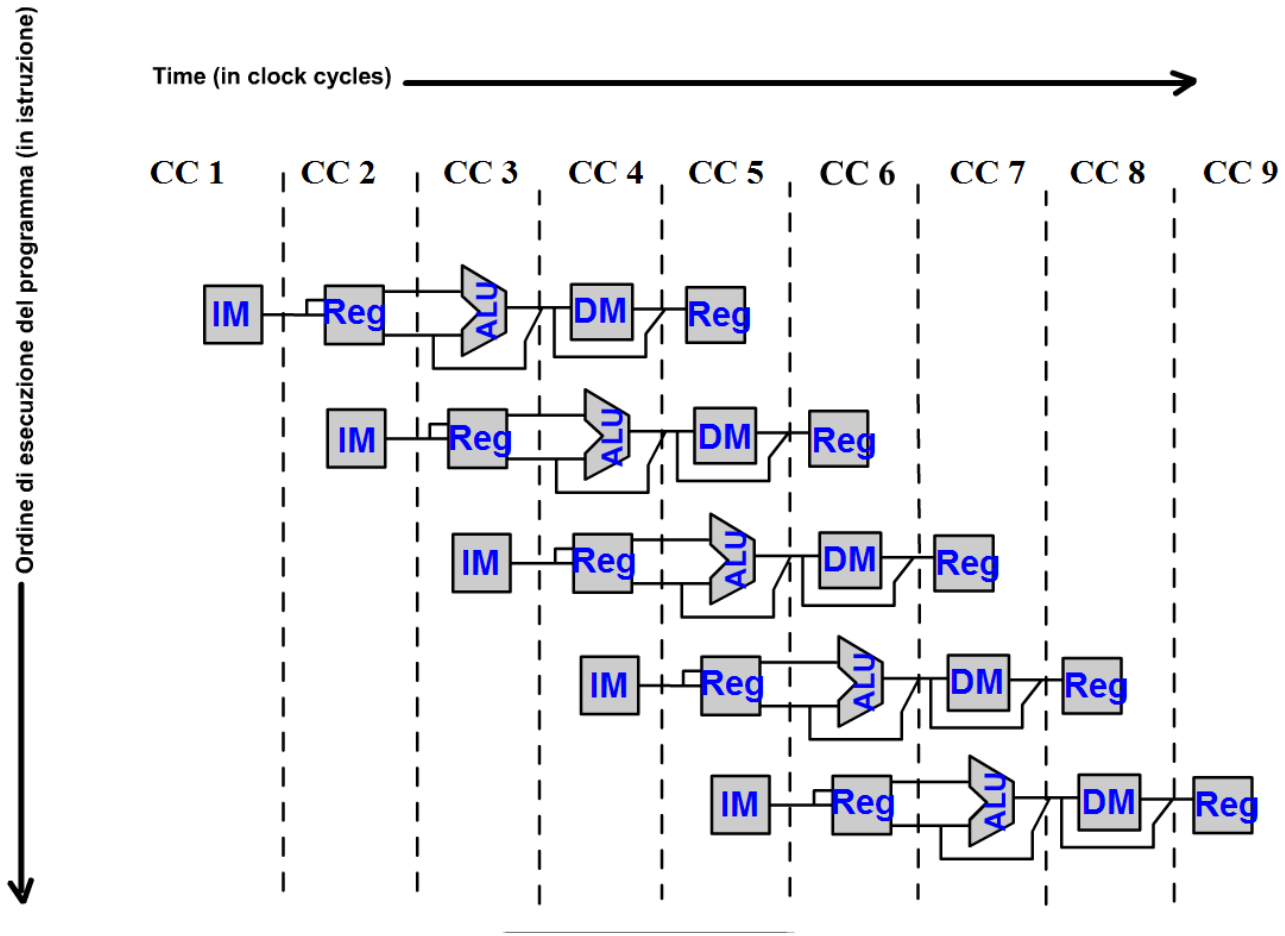
L'unità di controllo nella fase ID produce i segnali di controllo che verranno utilizzati negli stadi successivi EX, MM e WB

I segnali non utilizzati in uno stadio sono inoltrati agli stadi successivi.

Unità di Elaborazione + Unità di controllo



Risorse coinvolte



05_Risorse_coinvolte.exe

Problemi

Potenziali conflitti sulle risorse!
1. Con IM e DM separate, no conflict, ma a parità di ck cycle occorre una memoria 5X veloce!
2. Le fasi ID e WB usano lo stesso banco di registri nello stesso periodo di ck per istruzioni diverse!
3. Il PC va aggiornato ogni periodo di ck per fare un fetch/ck. Cosa accade per un branch il cui esito è noto soltanto nella fase di mem?

I registri A, B, Imm, nello stesso ck sono utilizzati nello stage di ex. (istruz. I), ma scritti nello stage di decode dall'istruz. I+1!
• Il registro IR viene scritto (nello stesso Ck) nello stage di IF (istruz. I) e il suo contenuto relativo all'istruz. I-4 serve nello stage WB!

Limiti alla esecuzione pipeline: i conflitti (Hazard)

Prof. G. Ascia

- **I conflitti (Hazard) impediscono che una istruzione venga eseguita nel ciclo di clock atteso**

-Structural hazards: Le risorse HW non supportano alcune combinazioni di istruzioni

-Data hazards: Un'istruzione dipende dal risultato di una istruzione che è ancora nella pipeline

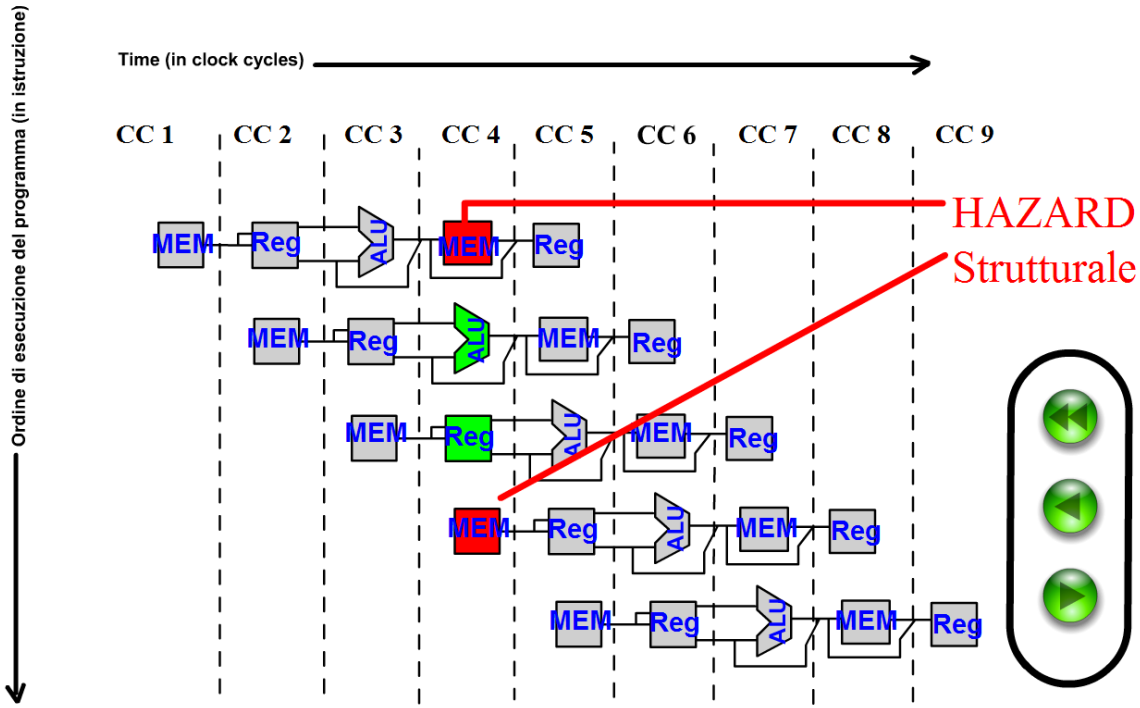
-Control hazards: Pipelining di branch e altre istruzioni che cambiano il PC

- **La soluzione più semplice è introdurre dei cicli di clck di stallo nella pipeline fino a quando l' hazard non è risolto, inserendo una o più "bolle" nella pipeline.**

Structural Hazard

Prof. G. Ascia

Esempio di structural hazard quando è presente un'unica memoria per le istruzioni e i dati



06_Structural_hazard_bubble.exe

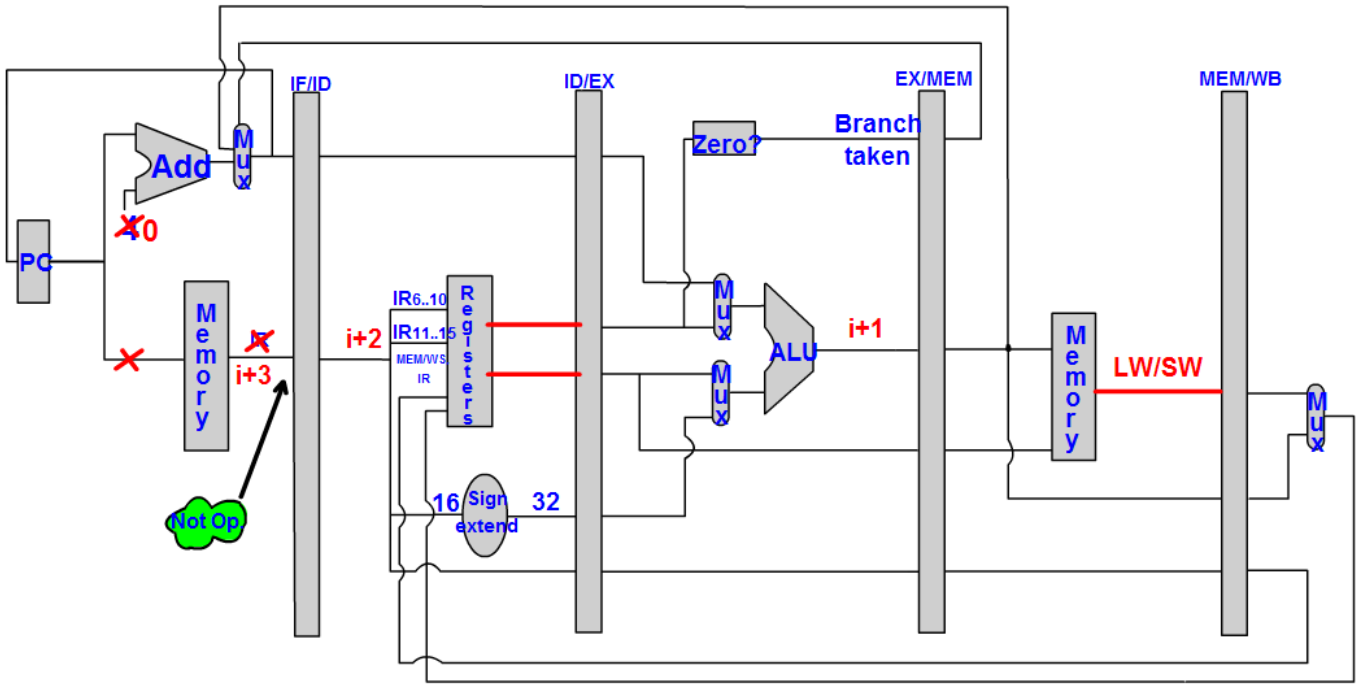
Structural Hazard

Soluzioni

- Introdurre una bolla, una nop (not operation), nella fase di ID e bloccare l'istruzione nella fase di Fetch
- Duplicare le risorse hardware

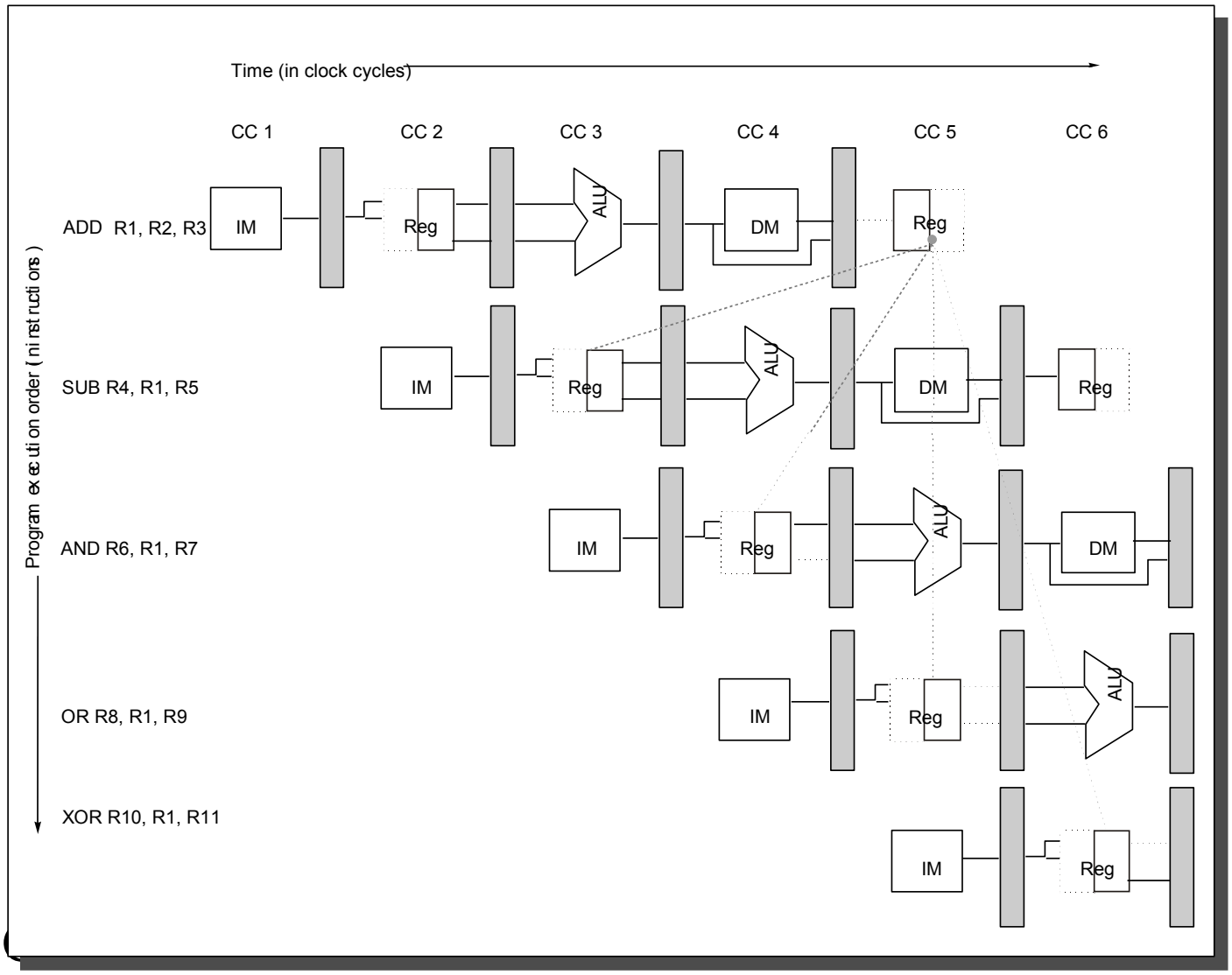
Structural Hazard: Stallo

Prof. G. Ascia

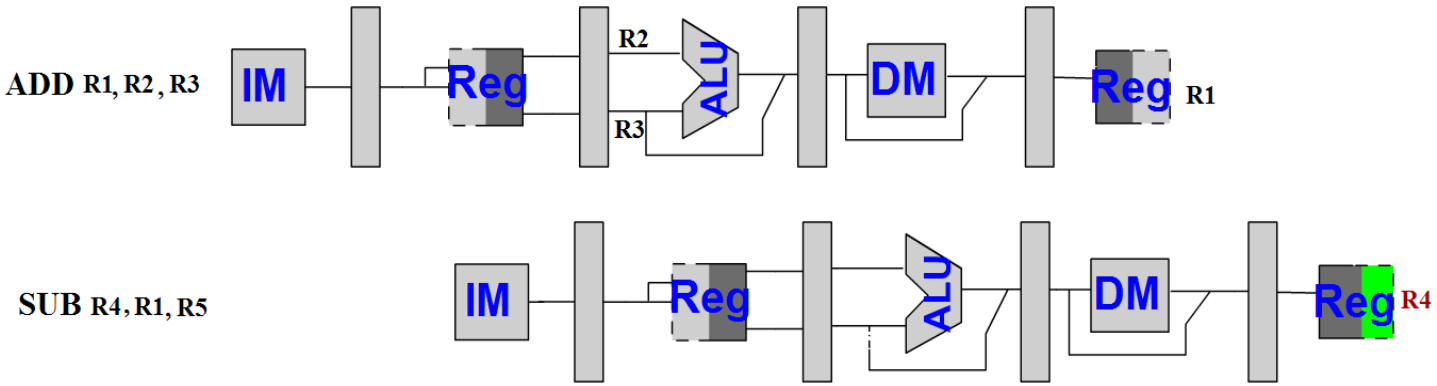


08_Structural_hazard_Implementazione_stallo.exe

Data Hazard


Time (in clock cycles)
CC 1
CC 2
CC 3
CC 4
CC 5
CC 6
Program execution order (in instructions)
ADD R1, R2, R3
SUB R4, R1, R5
AND R6, R1, R7
OR R8, R1, R9
XOR R10, R1, R11
IM
Reg
ALU
DM
Reg

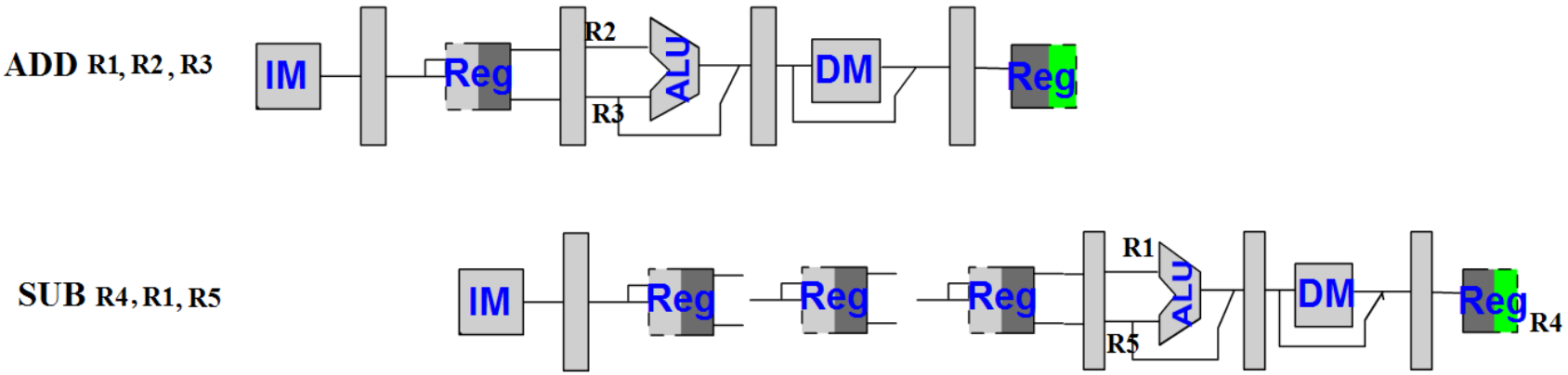
Data Hazard: esecuzione errata



09_Data_hazards_Esecuzione_errata.exe

Data Hazard: esecuzione corretta

Prof. G. Ascia



10_Data_hazards_Esecuzione_corretta-Introduzione_bolle.exe

Data Hazard: introduzione degli stalli

add r1, r2, r3	IF	ID	EX	MEM	WB			
sub r4, r1, r2		IF	ID	***stall***	***stall***	EX	MEM	WB

add r1, r2, r3	IF	ID	EX	MEM	WB			
subi r3, r2, 10		IF	ID	EX	MEM	WB		
addi r4, r1, 5			IF	ID	***stall***	EX	MEM	WB

Data Hazard: introduzione degli stalli

Una soluzione ai data hazard è l'introduzione di cicli di clock di stallo.

Poichè il data hazard viene scoperto nella fase ID, quando viene introdotto uno stallo per i data hazard:

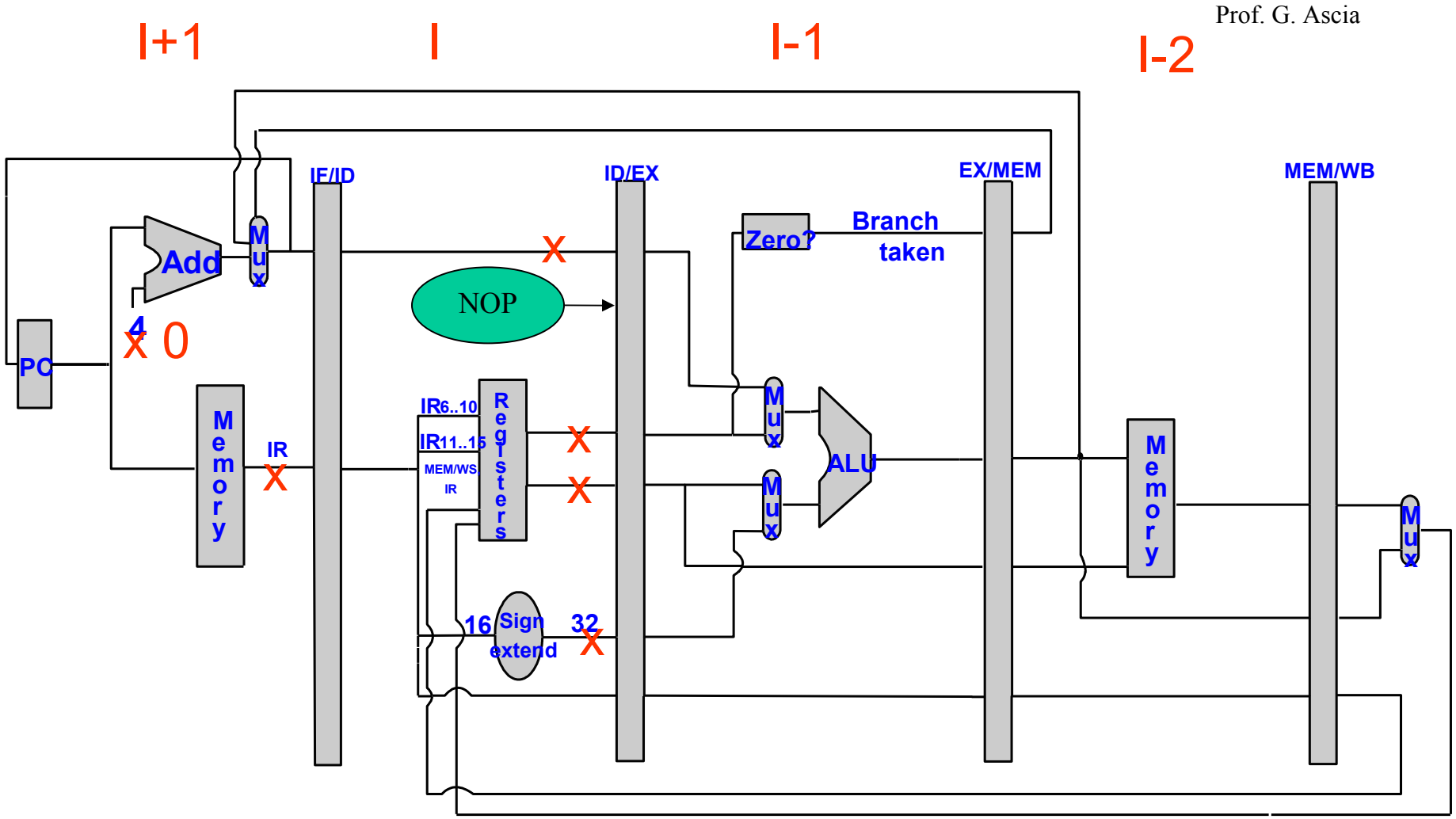
- Viene bloccata l'istruzione nella fase ID impedendo l'aggiornamento del registro IF/ID;
- Viene bloccata l'istruzione nella fase IF non aggiornando il PC
- Vengono scritti sul registro ID/EX i segnali di controllo relativi a una nop

I cicli di stallo vengono ripetuti fino a quando non viene aggiornato il registro destinazione

Il numero di cicli di clock di stallo dipende dalla distanza tra le istruzioni

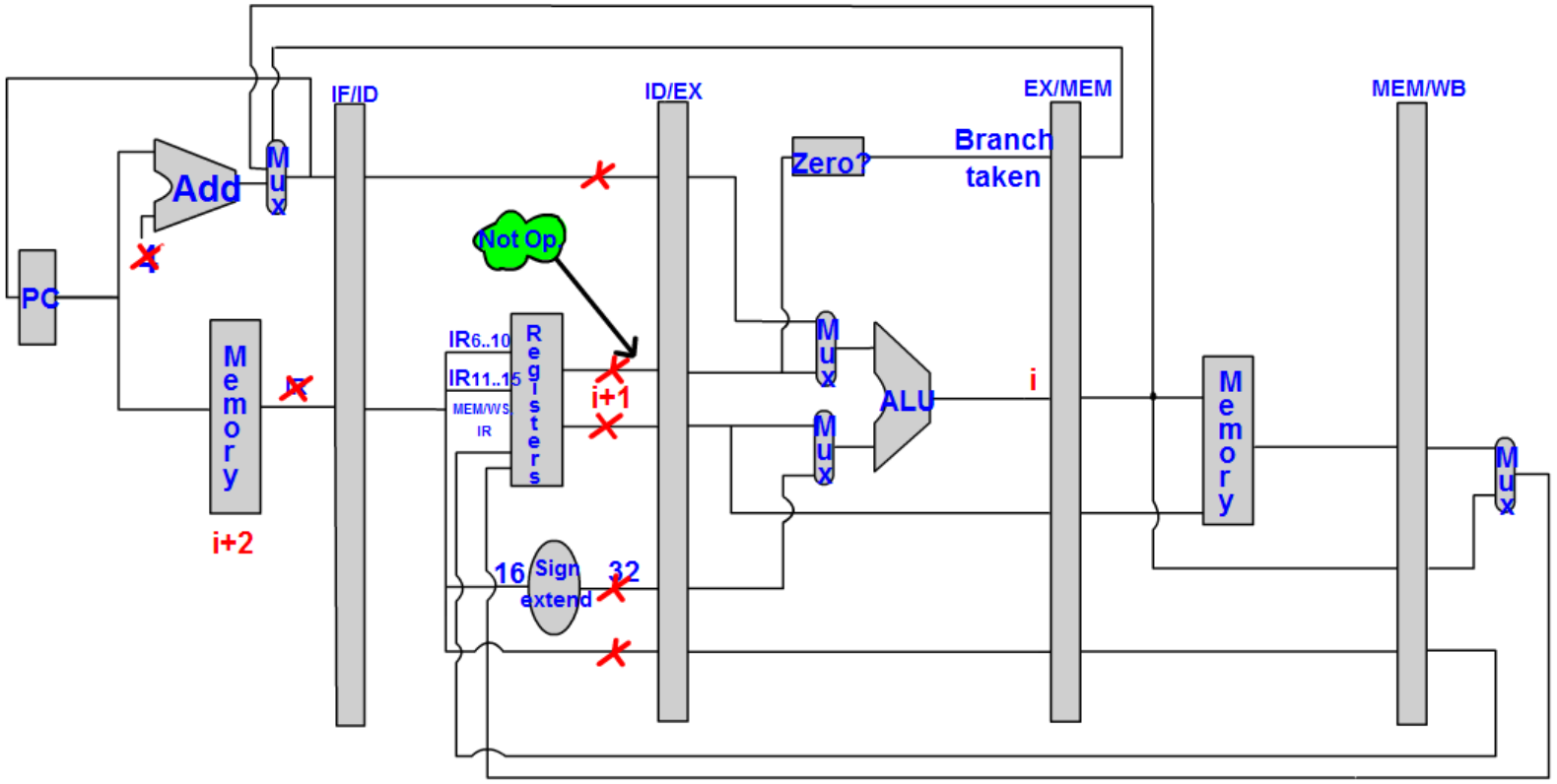
Introduzione stalli x data hazard

Prof. G. Ascia



Data Hazard: introduzione stalli

Prof. G. Ascia



11_Data_hazards_Datapath_Implementazione_introduzione_stalli.exe

Data Hazard: introduzione degli stalli


```
i    :   add r1, r2,r3
i+1:  sub r4, r1,r2
```

Supponendo che la scrittura in r1 avvenga nel primo semiperiodo del clock e la lettura nel secondo semiperiodo tra le istruzioni a distanza 1 sono necessari 2 cicli di clock di stallo

```
i    :   add r1, r2,r3
i+1:  subi r3, r2, 10
i+2:  addi r4, r1, 5
```

Supponendo che la scrittura in r1 avvenga nel primo semiperiodo del clock e la lettura nel secondo semiperiodo tra le istruzioni a distanza 2 sono necessari 1 cicli di clock di stallo

Data Hazard: Forwarding

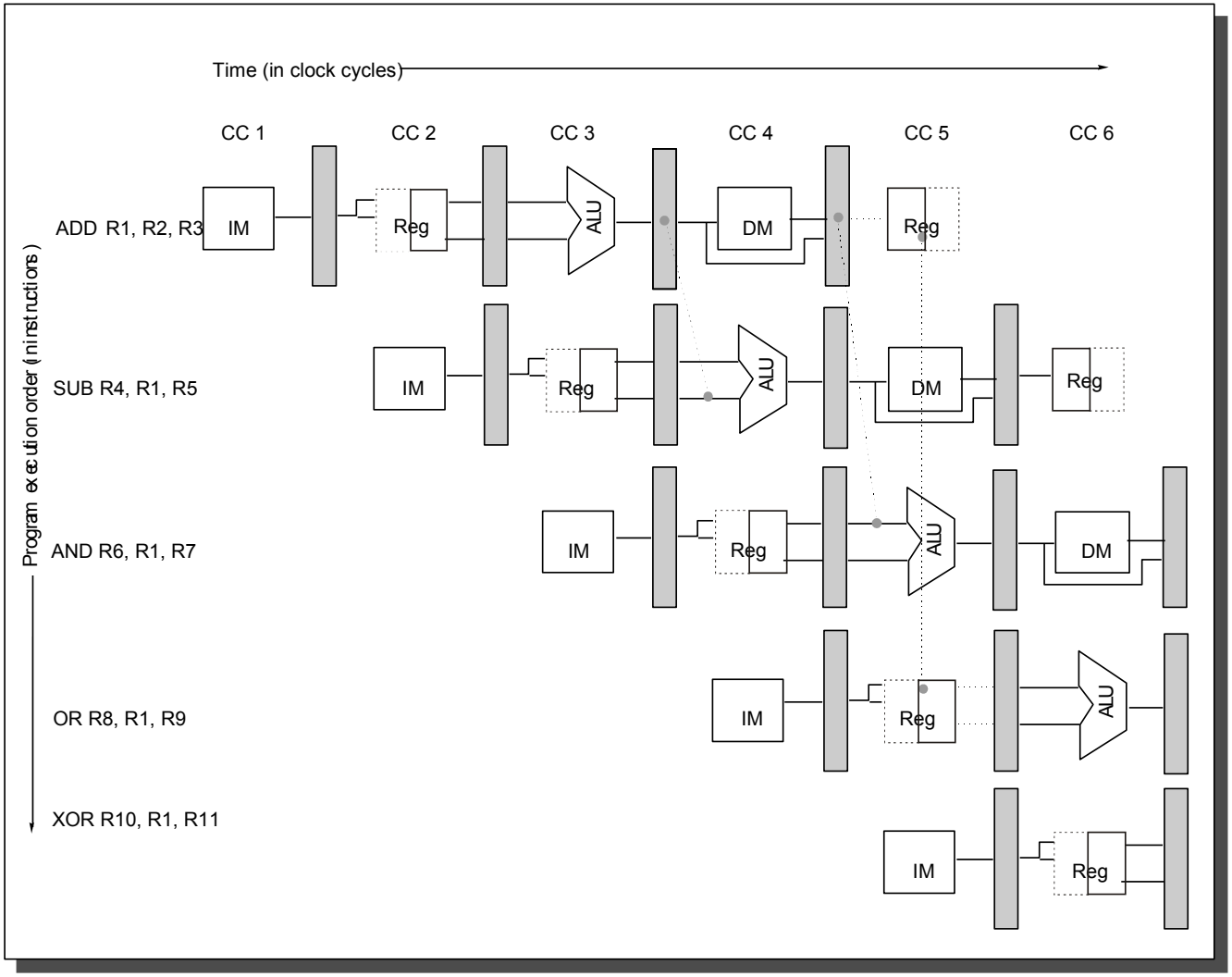
Prof. G. Ascia

Non si aspetta che il registro destinazione r_D sia stato aggiornato per fare avanzare nella fase di execute l'istruzione che ha bisogno del risultato

Il dato viene immediatamente utilizzato non appena è prodotto

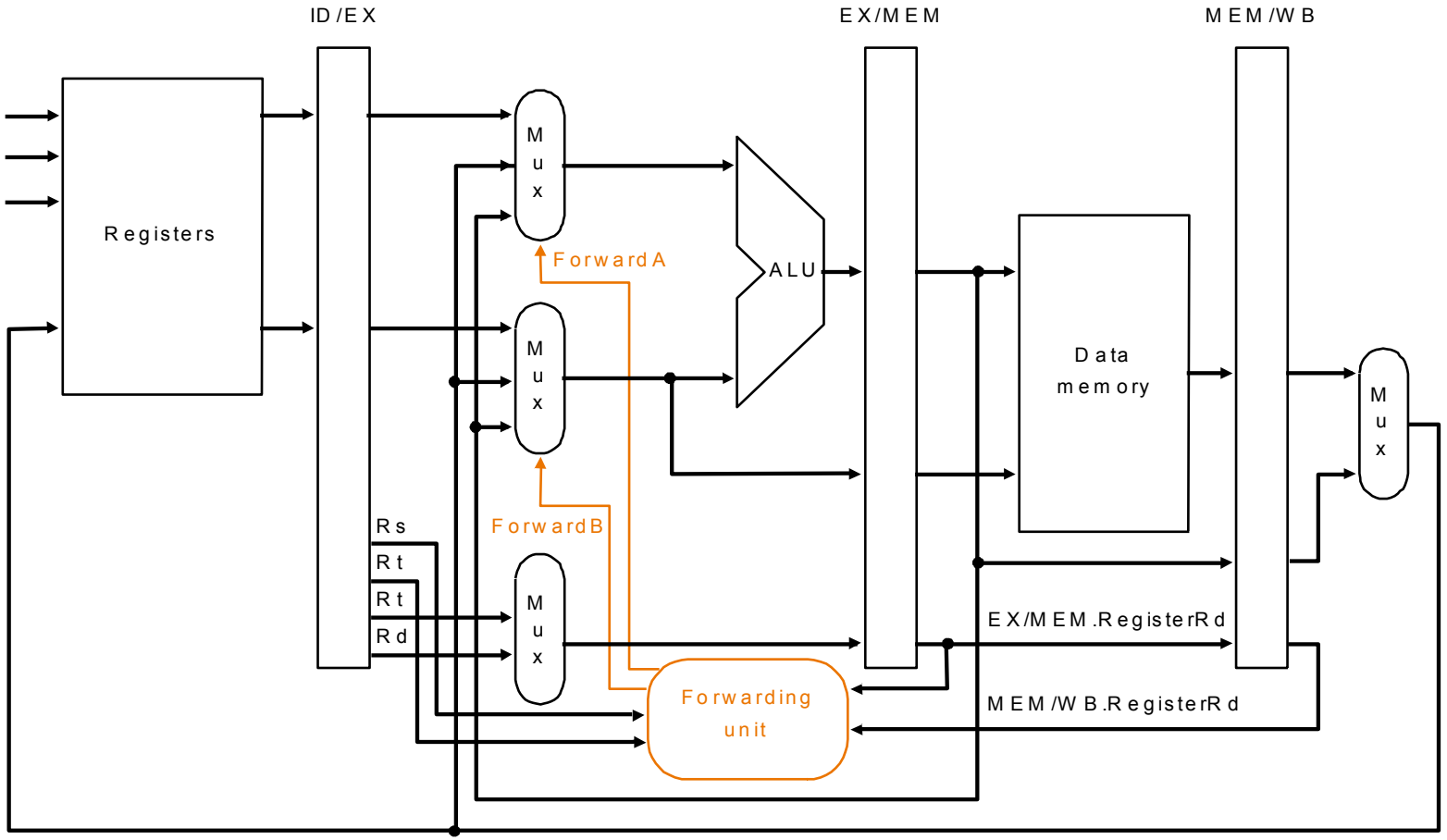
Forwarding per i Data Hazard

Prof. G. Ascia



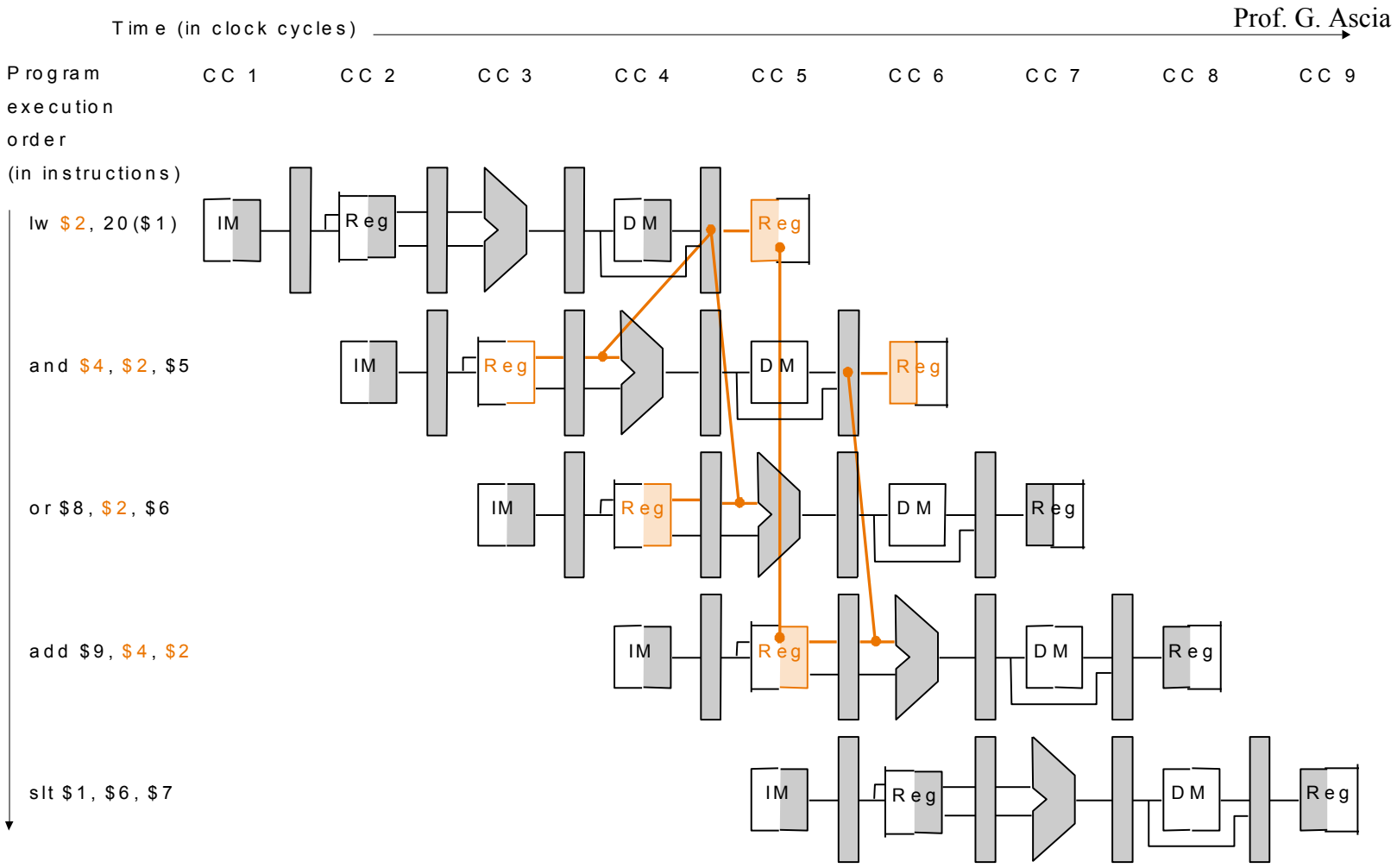
Implementazione del forwarding

Prof. G. Ascia

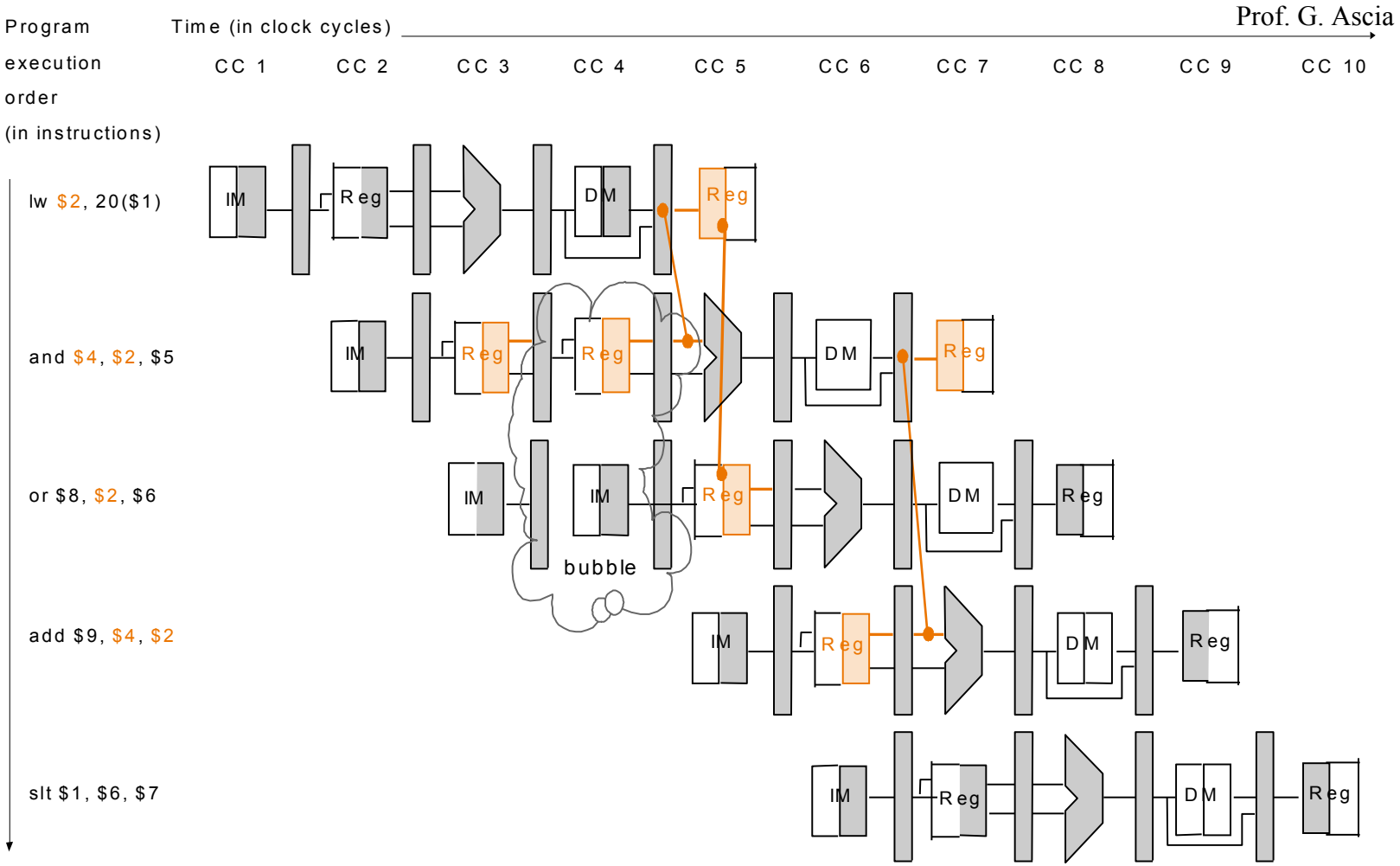


b. With forwarding

Forwarding per lw



Introduzione di uno stallo per lw



Software Scheduling

Try producing fast code for

a = b + c;

d = e – f;

assuming a, b, c, d ,e, and f in memory.

Slow code:

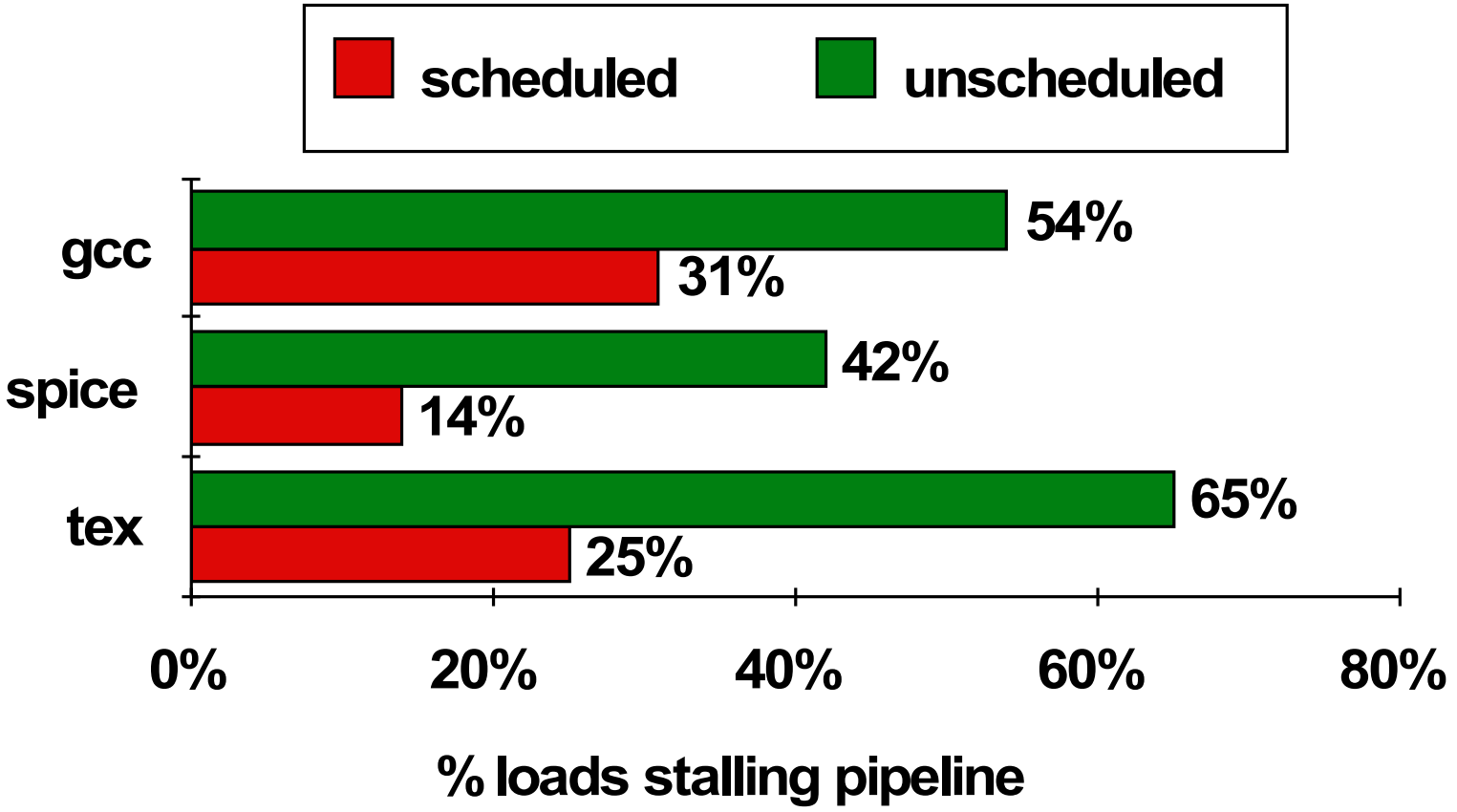
```
LW  Rb,b
LW  Rc,c
ADD     Ra,Rb,Rc
SW      a,Ra
LW  Re,e
LW  Rf,f
SUB     Rd,Re,Rf
SW d,Rd
```

Fast code:

```
LW  Rb,b
LW  Rc,c
LW  Re,e
ADD     Ra,Rb,Rc
LW  Rf,f
SW      a,Ra
SUB     Rd,Re,Rf
SW d,Rd
```

Compiler Avoiding Load Stalls

Prof. G. Ascia



Control Hazard

Prof. G. Ascia

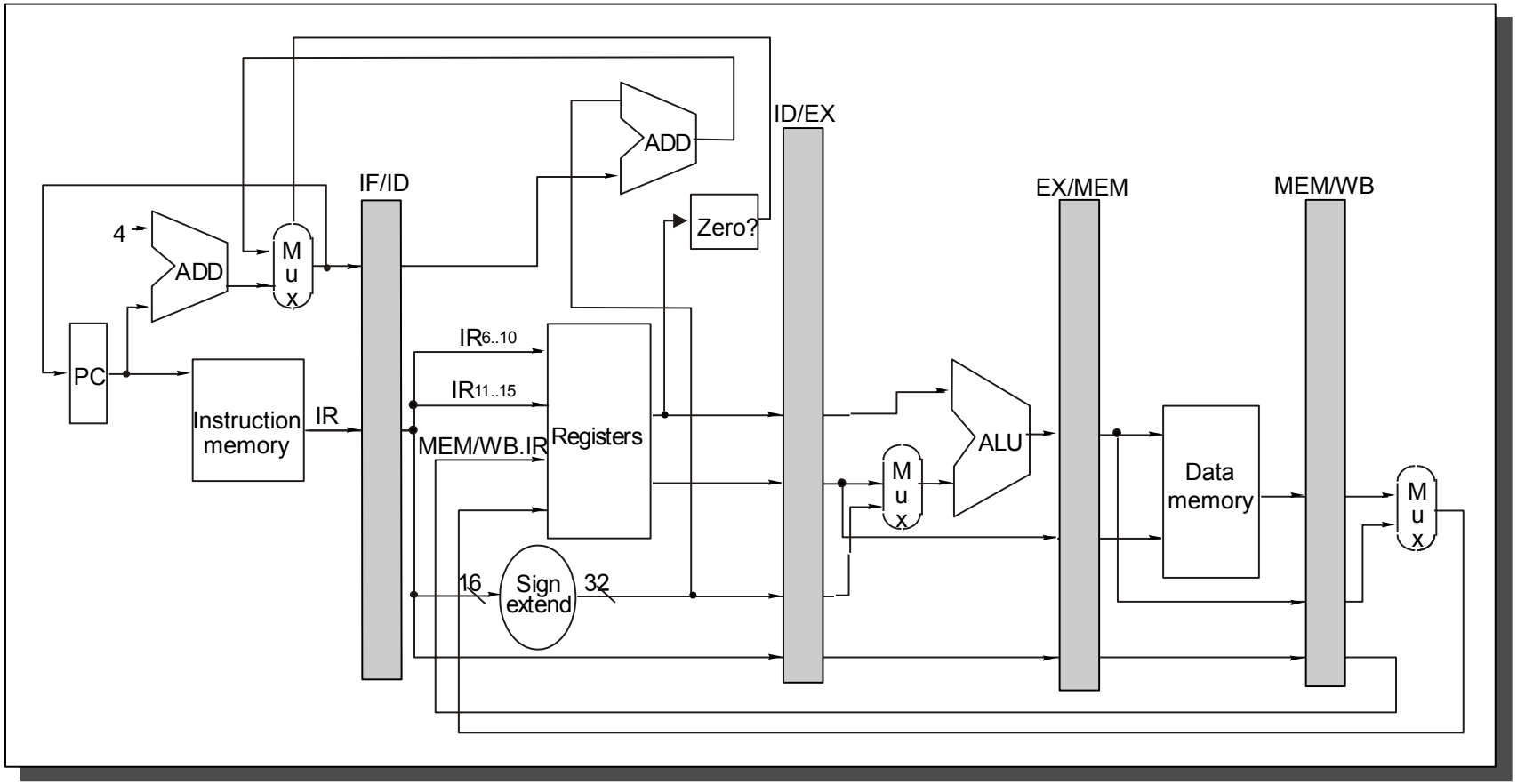
Un branch causa l'introduzione di 3 cicli di clock di stallo in attesa dell'esito del branch (l'aggiornamento del PC viene eseguito nella fase di MEM).

Branch instruction	IF	ID	EX	MEM	WB					
Branch successor		IF	***stall***	***stall***	***stall***	IF	ID	EX	MEM	WB
Branch successor + 1							IF	ID	EX	MEM
Branch successor + 2								IF	ID	EX
Branch successor + 3									IF	ID
Branch successor + 4										IF
Branch successor + 5										

Control Hazard

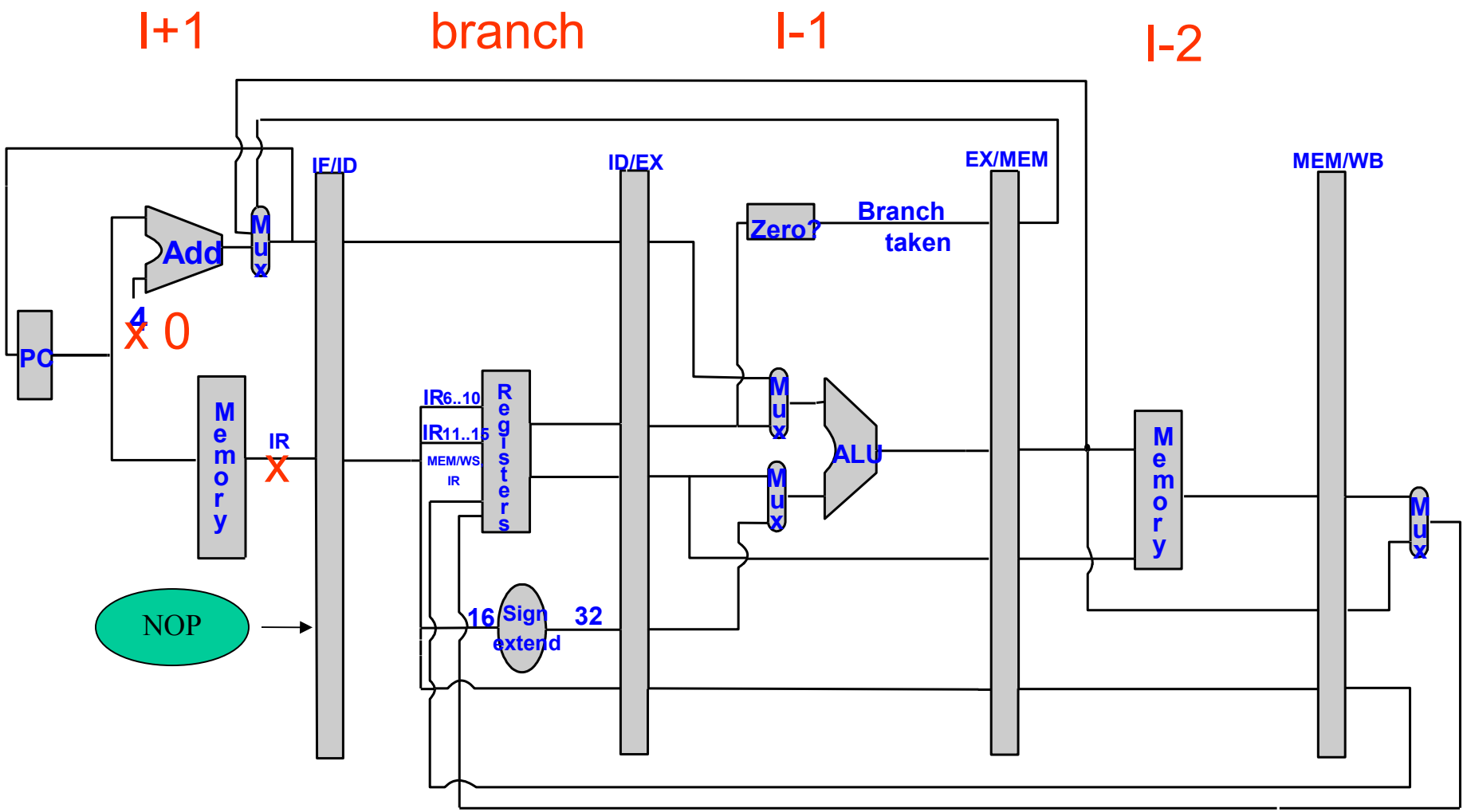
Prof. G. Ascia

Il numero di cicli può essere ridotto a un solo ciclo anticipando la verifica nello stadio ID



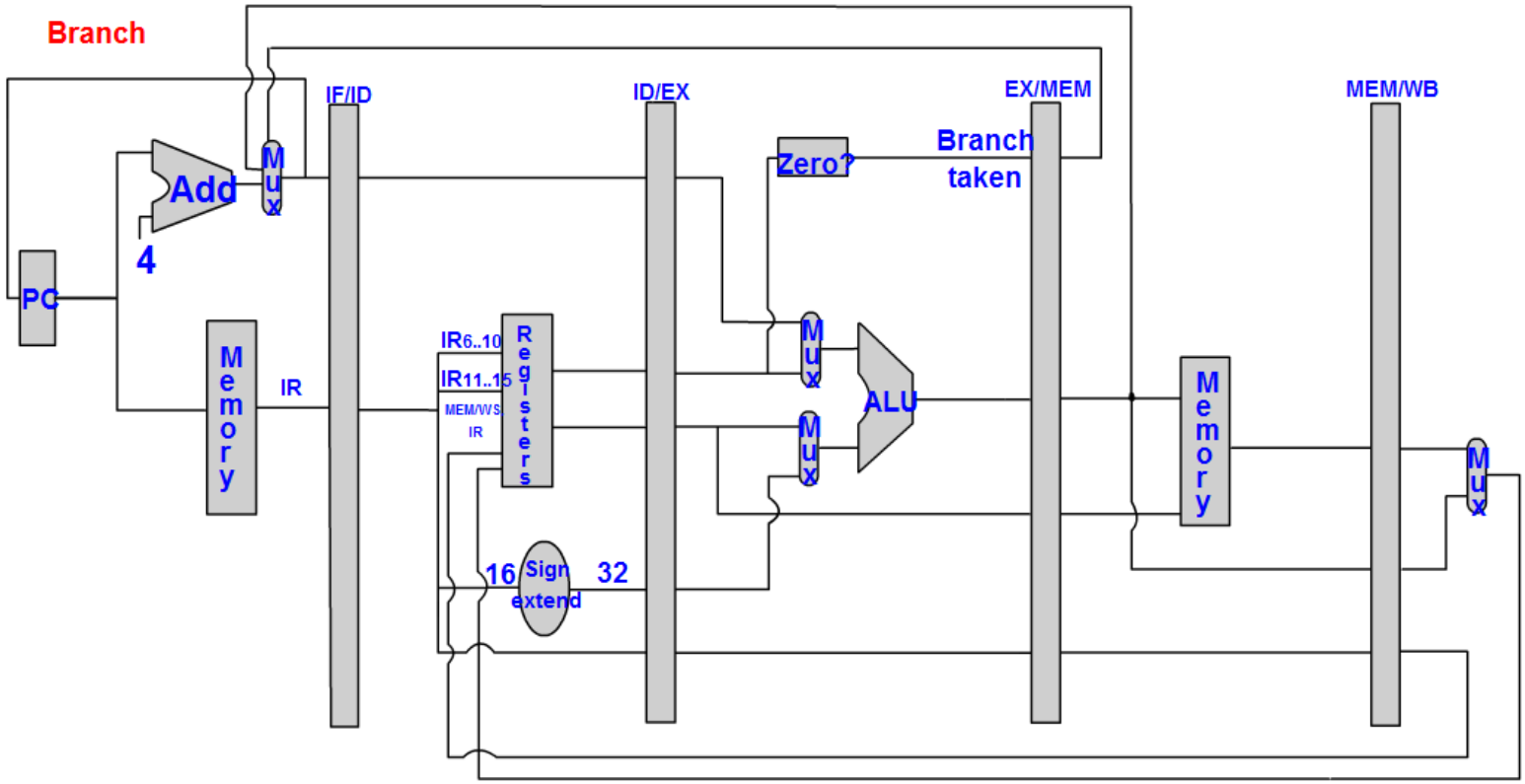
Introduzione stalli x control hazard

Prof. G. Ascia



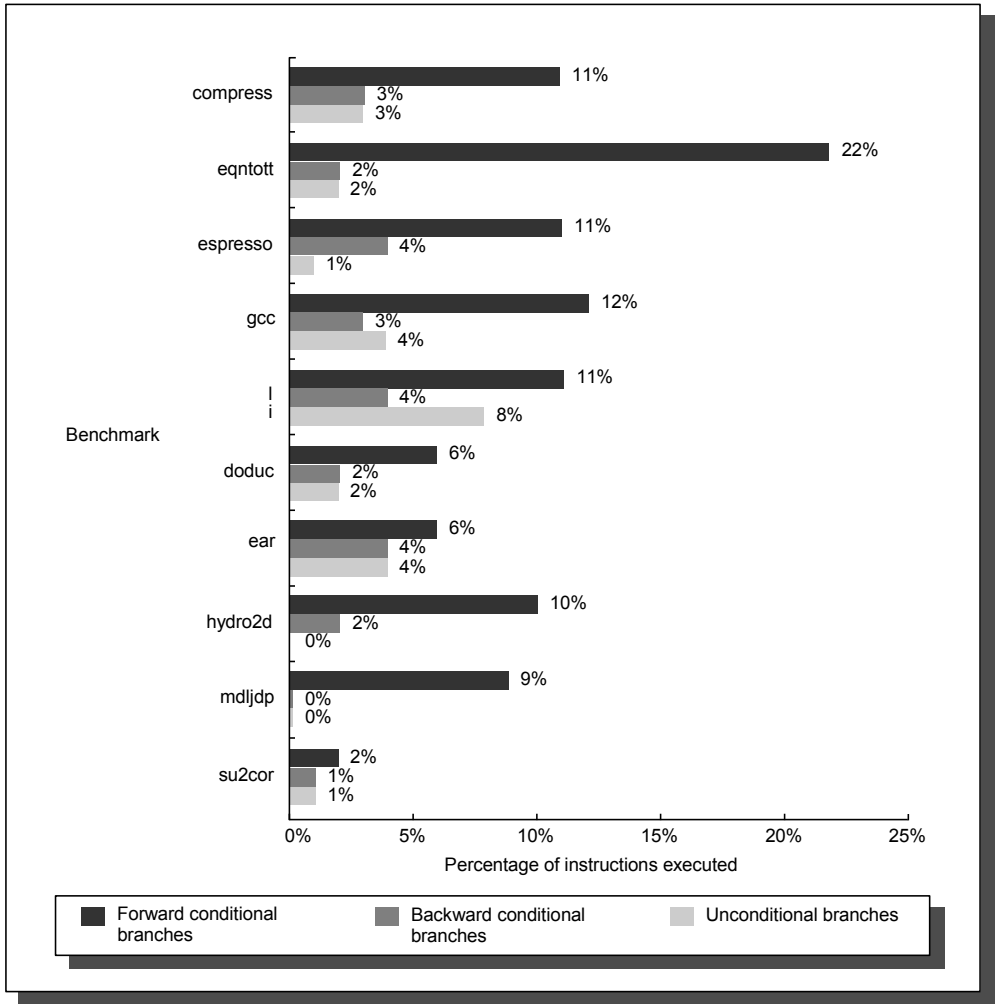
Control Hazard: stalli

Prof. G. Ascia

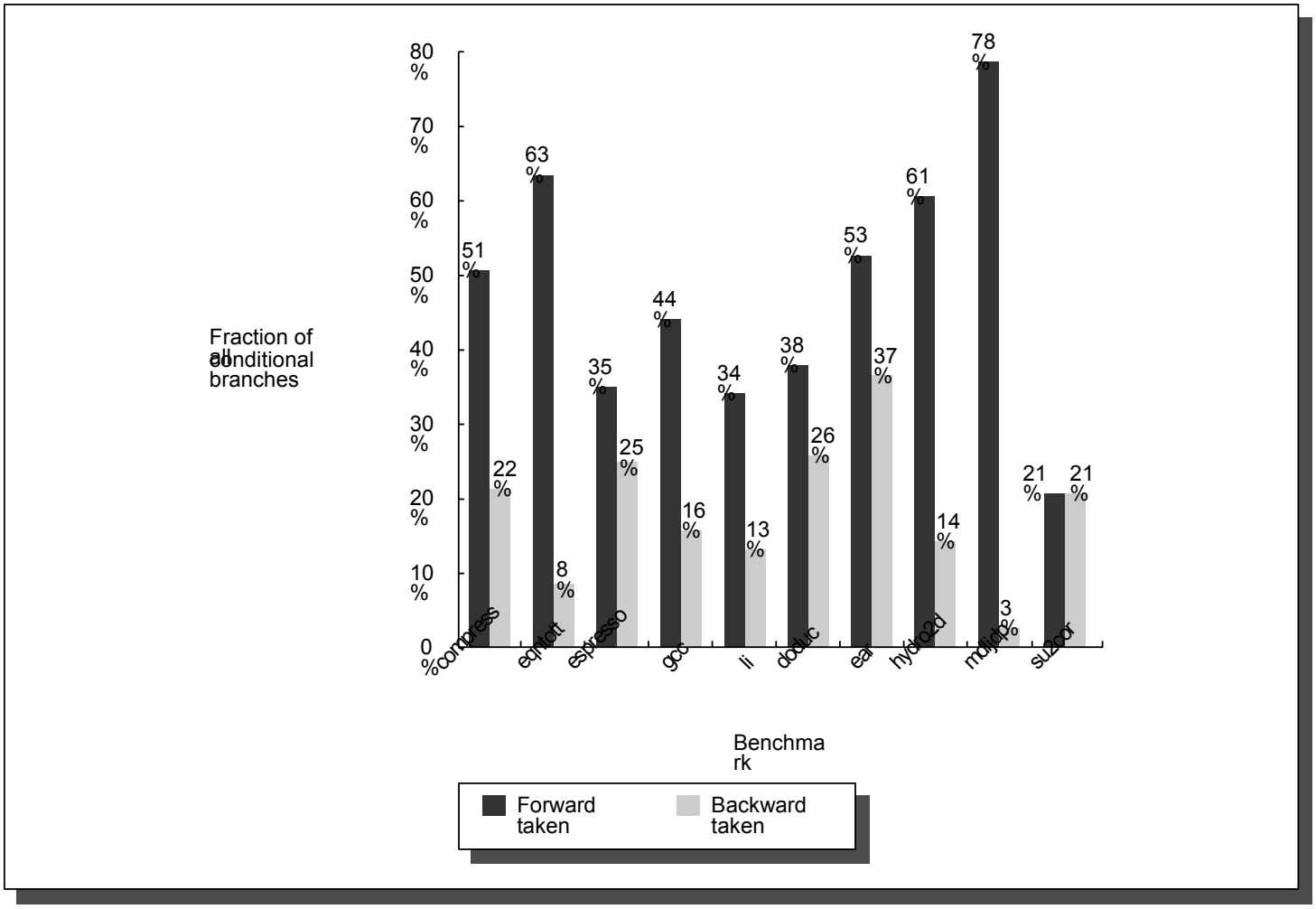


12_Control_hazard_Datapath_Implementazione_introduzione_stalli.exe

Frequenza branch



Frequenza branch



Four Branch Hazard Alternatives

#1: Stall until branch direction is clear

#2: Predict Branch Not Taken

- **Execute successor instructions in sequence**
- **"Squash" instructions in pipeline if branch actually taken**
- **Advantage of late pipeline state update**
- **47% DLX branches not taken on average**
- **PC+4 already calculated, so use it to get next instruction**

#3: Predict Branch Taken

- **53% DLX branches taken on average**
- **But haven't calculated branch target address in DLX**
- **DLX still incurs 1 cycle branch penalty**
- **Other machines: branch target known before outcome**

Control Hazard

The predict-not-taken scheme and the pipeline sequence when the branch is untaken (top) and taken (bottom)

Untaken branch instr.	IF	ID	EX	MEM	WB			
Instruction i+1		IF	ID	EX	MEM	WB		
Instruction i+2			IF	ID	EX	MEM	WB	
Instruction i+3				IF	ID	EX	MEM	WB
Instruction i+4					IF	ID	EX	MEM WB

Taken branch instr.	IF	ID	EX	MEM	WB			
Instruction i+1		IF	**idle**	**idle**	**idle**	**idle**		
Branch target			IF	ID	EX	MEM	WB	
Branch target + 1				IF	ID	EX	MEM	WB
Branch target + 2					IF	ID	EX	MEM WB

Four Branch Hazard Alternatives

Prof. G. Ascia

#4: Delayed Branch

- **Define branch to take place AFTER a following instruction**

```
branch instruction
 sequential successor1
 sequential successor2
 ........
 sequential successorn
branch target if taken
```

- **1 slot delay allows proper decision and branch target address in 5 stage pipeline**
- **DLX uses this**

Control Hazard-Delay slot

The behavior of a delayed branch is the same wheter or not the branch is taken.

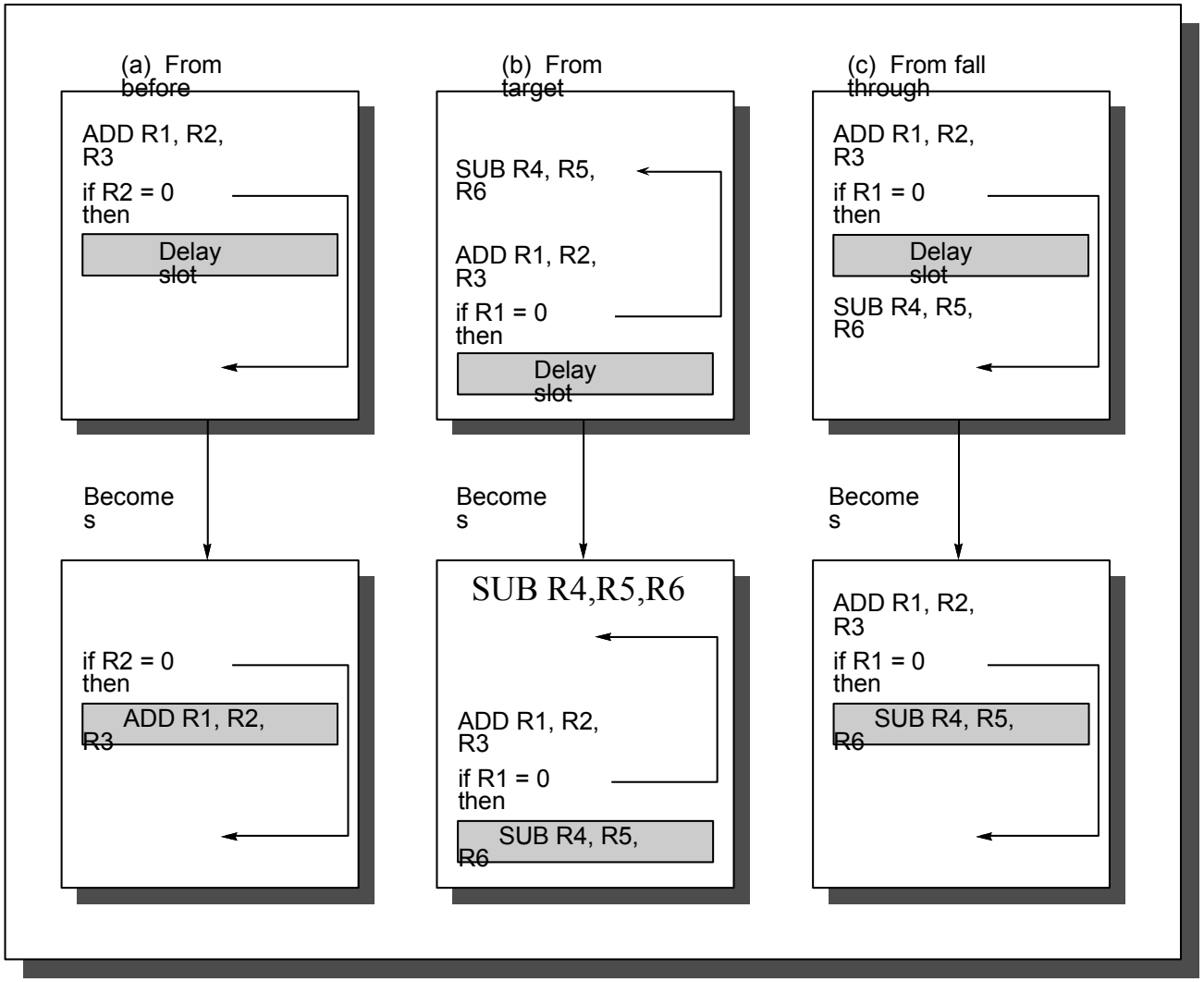
Prof. G. Ascia

Untaken branch instr.	IF	ID	EX	MEM	WB			
Branch-delay instr. (i+1)		IF	ID	EX	MEM	WB		
Instruction i+2			IF	ID	EX	MEM	WB	
Instruction i+3				IF	ID	EX	MEM	WB
Instruction i+4					IF	ID	EX	MEM WB

Taken branch instr.	IF	ID	EX	MEM	WB			
Branch-delay instr. (i+1)		IF	ID	EX	MEM	WB		
Branch target			IF	ID	EX	MEM	WB	
Branch target + 1				IF	ID	EX	MEM	WB
Branch target + 2					IF	ID	EX	MEM WB

Control Hazard-Delay slot

Prof. G. Ascia



Delayed Branch

- **Where to get instructions to fill branch delay slot?**
  - **Before branch instruction**
  - **From the target address: only valuable when branch taken**
  - **From fall through: only valuable when branch not taken**
  - **Cancelling branches allow more slots to be filled**

- **Compiler effectiveness for single branch delay slot:**
  - **Fills about 60% of branch delay slots**
  - **About 80% of instructions executed in branch delay slots useful in computation**
  - **About 50% (60% x 80%) of slots usefully filled**

Pipeline performance con Hazard
Speedup rispetto alla versione sequenziale

Prof. G. Ascia

$$CPU_{TIME} = IC \cdot CPI \cdot TC$$

$$Speedup = \frac{CPU_{TIME\ Sequenziale}}{CPU_{TIME\ Pipeline}} = \frac{CPI_{Seq} \cdot T_{CK\ Seq}}{CPI_{Pipe} \cdot T_{CK\ Pipe}}$$

$$CPI_{Pipe} = CPI_{Ideale} + Nr_{\text{Cicli di clock di stallo per istruzione}}$$

Hp: Stage bilanciati, no overhead x pipeline, $T_{CK\ Seq} = T_{CK\ Pipe}$

$$Speedup = \frac{CPI_{Seq}}{1 + Nr_{\text{Cicli Stallo per Istruzione}}} = \frac{\text{Stadi della Pipe}}{1 + Nr_{\text{Cicli Stallo per Istruzione}}}$$

Pipeline performance: esempio1 (1/2)

Dato un programma con seguente instruction mix

- Lw 20%
- Sw 10%
- Branch 20%
- ALU 50 %

ipotizzando di avere un'unica memoria, utilizzare il forwarding, calcolare l'esito del branch nello stadio di Decode e che il 50% delle Lw è seguito da un'istruzione che da essa dipende

calcolare lo speedup dell'architettura pipeline rispetto a una sequenziale.

$$CPI_{Pipe}=CPI_{ideale}+Nr_{CicliStalloPerIstruzione}=$$

$$=1+f_{StructHazard}*Stalli_{StructHazard}+ f_{Branch}*Stalli_{Branch}+ f_{DataHazard\_Lw}*Stalli_{DataHazard\_Lw}$$

Pipeline performance: esempio1 (2/2)

Poiché

- $f_{StructHazard} = f_{Lw} + f_{Sw} = 0,2+0,1=0,3$ $Stalli_{StructHazard} = 1$
- $f_{Branch} = 0,2$ $Stalli_{Branch} = 1$
- $f_{DataHazard\_Lw} = 20\%*50\% = 0,1$ $Stalli_{DataHazard\_Lw} = 1$

$CPI_{Pipe} = 1 + 0,3*1+0,2*1+0,1*1 = 1+0,6 = 1,6$

$$\text{Speedup} = \frac{CPI_{Seq}}{CPI_{Pipe}} = \frac{\text{Stadi della Pipe}}{1 + Nr_{\text{Cicli Stallo per Istruzione}}} = \frac{5}{1,6} = 3,125$$

Pipeline performance: esempio2 (1/2)

Dato un programma con seguente instruction mix

- Lw 25%
- Sw 15%
- Branch 20%
- ALU 40 %

ipotizzando di avere due memorie, di non utilizzare il forwarding, di calcolare l'esito del branch nello stadio di MemAccess, che il 30% delle istruzioni ALU ha una dipendenza dati con una istruzione a distanza 1 e il 10% delle istruzioni ALU ha una dipendenza dati con una istruzione a distanza 2,

calcolare lo speedup dell'architettura pipeline rispetto a una sequenziale.

- $CPI_{Pipe}=CPI_{ideale}+Nr_{CicliStalloPerIstruzione}=$

$=1+f_{Branch}*Stalli_{Branch}+ f_{DataHazard\_ALU\_Distanza1}*Stalli_{DataHazard\_ALU\_Distanza1}+ f_{DataHazard\_ALU\_Distanza2}*Stalli_{DataHazard\_ALU\_Distanza1}$

Pipeline performance: esempio2 (2/2)

Poiché

- f_{Branch} =0,2 $Stalli_{Branch}$ =3
- $f_{DataHazard\_ALU\_Distanza1}$=40%*30%=0,12 $Stalli_{DataHazard\_ALU\_Distanza1}$ =2
- $f_{DataHazard\_ALU\_Distanza2}$ =40%*10%=0,04 $Stalli_{DataHazard\_ALU\_Distanza2}$ =1

CPI_{Pipe}= 1+ 0,2*3+0,12*2+0,04*1=1+0,6+0,24+0,04=1,88

$$\text{Speedup} = \frac{CPI_{Seq}}{CPI_{Pipe}} = \frac{\text{Stadi della Pipe}}{1 + Nr_{\text{Cicli Stallo per Istruzione}}} = \frac{5}{1{,}88} = 2{,}66$$