Architettura degli elaboratori
Esercitazione

UniVR - Dipartimento di Informatica

Fabio Irimie

2° Semestre 2023/2024

Indice

1 Esercizi svolti

1.1 Esercizio 1

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.

Istruzione	1	2	3	4	5	6	7	8	9	10	11
addl %eax, %ebx	F	D	E	M	S						
movl $4, %ecx		F	D	E	M	S					
subl %ebx, %ecx			F	D	D	D	D	M	S		
movl $4, %edx				F	F	F	F	D	E	M	S

1.2 Esercizio 2

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1.

Istruzione	1	2	3	4	5	6	7	8	9	10	11
ciclo: addl %eax, %ebx	F	D	E	M	S		F	D	D	E	M
movl $4, %ecx		F	D	E	M	S		F	F	D	E
subl %eax, %edx			F	D	E	M	S			F	D
movl $6, %ebx				F	D	E	M	S			F
jmp ciclo					F	D	E	M	S		

1.3 Esercizio 3

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1. Si ipotizzi che il salto avvenga. Si ignorino le tecniche del Delay Slot e della Branch Prediction. I commenti #yes e #no indicano se il salto avviene o meno.

Istruzione	1	2	3	4	5	6	7	8	9	10	11	12	13
inizio: inc %ebx	F	D	E	M	S				F	D	E	M	S
movl %ecx, %edx		F	D	E	M	S				F	D	E	M
cmpl %eax, 0x86FF			F	D	E	M	S				F	D	E
jne inizio #yes				F	D	D	D	D	E	M	S	F	D
movl %ecx, %edx					F	F	F	F					F

1.4 Esercizio 4

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1 e che il salto non avvenga.

Istruzione	1	2	3	4	5	6	7	8	9	10	11
START: subl %eax, %ebx	F	D	E	M	S						
jz START #no		F	D	D	D	D	E	M	S		
subl %ebx, %ecx			F	F	F	F	D	E	M	S	
movl %edx, %eax							F	D	E	M	S

1.5 Esercizio 5

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1 e che il salto non avvenga.

Istruzione	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ciclo: addl %eax, %ebx	F	D	E	M	S										
movl %edx, %ecx		F	D	E	M	S									
subl %ebx, %ecx			F	D	D	D	D	E	M	S					
jz ciclo #no				F	F	F	F	D	D	D	D	E	M	S	
movl %ecx, %edx								F	F	F	F	D	E	M	S

1.6 Esercizio 6

Si consideri una CPU con una pipeline a 4 stadi (F, D, E, W). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi che la pipeline sia vuota al tempo 1 e che jz faccia riferimento all'istruzione subl.

Istruzione	1	2	3	4	5	6	7	8	9	10	11	12	13
ciclo: addl %eax, %ebx	F	D	E	W									
movl %ebx, %ecx		F	D	D	D	E	W						
subl %eax, %ecx			F	F	F	D	D	D	E	W			
jz ciclo #no						F	F	F	D	D	D	E	W

1.7 Esercizio 7

Si consideri una CPU con una pipeline a 5 stadi (F, D, E, M, S). Si riporti nel seguente diagramma, per ogni istruzione, lo stadio della pipeline coinvolto in ogni istante di clock. Si ipotizzi la pipeline vuota al tempo 1 e si facciano le opportune ipotesi sul salto condizionale.
Nel caso in cui il salto non viene effettuato il diagramma sarà il seguente:

Istruzione	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
init: movl %ecx, %edx	F	D	E	M	S										
addl $4, %ebx		F	D	E	M	S									
cmpl 0x319FA, %ebx			F	D	D	D	D	E	M	S					
jnz init #no				F	F	F	F	D	D	D	D	E	M	S	
addl %eax, %ecx								F	F	F	F	D	E	M	S

Nel caso in cui il salto viene effettuato il diagramma sarà il seguente:

Istruzione	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
init: movl %ecx, %edx	F	D	E	M	S							F	D	E	M	S		
addl $4, %ebx		F	D	E	M	S							F	D	E	M	S	
cmpl 0x319FA, %ebx			F	D	D	D	D	E	M	S				F	D	D	D	D
jnz init #yes				F	F	F	F	D	D	D	D	E	M	S	F	F	F	F
addl %eax, %ecx								F	F	F	F							

2 Esercizi moodle

2.1 Esercizio 1

Una CPU con una pipeline a 2 stadi viene sostituita con una CPU con una pipeline a 4 stadi. Se il tempo totale di esecuzione di una singola istruzione è rimasto invariato, qual è il minimo ed il massimo incremento delle prestazioni che si può attendere?

Il massimo di incremento delle prestazioni è di 2 perchè raddoppiando il numero di stadi si raddoppiano le prestazioni. Il minimo di incremento delle prestazioni è di 1, quindi non si hanno miglioramenti.

2.2 Esercizio 2

Per il seguente esercizio che coinvolge 4 istruzioni successive eseguite da una CPU con pipeline a 4 stadi, individuare lo stadio in cui si trova ogni istruzione ad ogni ciclo macchina (dove qui consideriamo un massimo di 11 cicli coinvolti). Dopodichè ripetere gli esercizi introducendo le migliorie architetturali discusse a lezione al fine di ridurre gli stalli nella pipeline. Per migliorie architetturali parliamo di:

- Inoltro di operandi (sia per operazioni logico-aritmetiche che di memoria)
- Branch prediction (o delay slot)

L'inoltro può avvenire su istruzioni a distanza maggiore di 1, naturalmente. E' evidente che la dipendenza di dati si può avere, in questa pipeline a 4 stadi, tra istruzioni fino a distanza 4. Ognuno di questi casi va identificato (in software o hardware) e prevenuto/risolto onde evitare esecuzione scorretta.

Istruzione	1	2	3	4	5	6	7	8	9	10	11
loop: addl %eax, %ebx	F	D	E	W							
movl ind %ecx		F	D	E	W						
subl %ebx, %ecx			F	D	D	D	E	W			
jz loop				F	F	F	D	D	D	E	W

Ottimizzando con la branch prediction si ha:

Istruzione	1	2	3	4	5	6	7	8	9	10	11
loop: addl %eax, %ebx	F	D	E	W					F	D	E
movl ind %ecx		F	D	E	W					F	D
subl %ebx, %ecx			F	D	D	D	E	W			F
jz loop				F	F	F	D	D	D	E	W

2.3 Esercizio 3

Per il seguente esercizio che coinvolge 4 istruzioni successive eseguite da una CPU con pipeline a 4 stadi, individuare lo stadio in cui si trova ogni istruzione ad ogni ciclo macchina (consideriamo un massimo di 11 cicli coinvolti). Dopodichè ripetere gli esercizi introducendo le migliorie architetturali discusse a lezione al fine di ridurre gli stalli nella pipeline.

Nel caso in cui il ciclo non viene effettuato si ha:

Istruzione	1	2	3	4	5	6	7	8	9	10	11
inizio: mull %eax, %ebx	F	D	E	W							F
movl %ecx ind		F	D	E	W						
cmpl %ebx, 0h			F	D	D	E	W				
jnz inizio				F	F	D	D	D	E	W	

Ottimizzando con branch prediction si ha:

Istruzione	1	2	3	4	5	6	7	8	9	10	11
inizio: mull %eax, %ebx	F	D	E	W				F	D	E	W
movl %ecx ind		F	D	E	W				F	D	E
cmpl %ebx, 0h			F	D	D	E	W			F	D
jnz inizio				F	F	D	D	D	E	W	F

2.4 Esercizio 4

Per il seguente esercizio che coinvolge 5 istruzioni successive eseguite da una CPU con pipeline a 4 stadi, individuare lo stadio in cui si trova ogni istruzione ad ogni ciclo macchina. Dopodichè ripetere l'esercizio introducendo le migliorie a livello hardware o di compilatore discusse a lezione al fine di ridurre gli stalli nella pipeline.

Istruzione	1	2	3	4	5	6	7	8	9	10	11	12	13	14
subl $2, %ebx	F	D	E	W										
movl ind, %ecx		F	D	E	W									
addl %eax, %ecx			F	D	D	D	E	W						
movl ind, %ebx				F	F	F	D	E	W					
movl %ecx, ind							F	D	D	E	W			

2.5 Esercizio 5

Per il seguente esercizio che coinvolge 6 istruzioni eseguite da una CPU con pipeline a 4 stadi, individuare lo stadio in cui si trova ogni istruzione ad ogni ciclo macchina ipotizzando di predire che il salto non avvenga e che tale predizione sia confermata. Si assume che l'istruzione successiva al salto sia conosciuta al ciclo immediatamente successivo (perchè usiamo predizione statica oppure perchè l'istruzione di salto si trova nel buffer di predizione), nonchè che la predizione risulti corretta. Dopodichè ripetere l'esercizio introducendo le migliorie a livello hardware o di compilatore discusse a lezione al fine di ridurre gli stalli nella pipeline.

Istruzione	1	2	3	4	5	6	7	8	9	10	11	12	13	14
j1: addl %ebx, %ecx	F	D	E	W										
movl ind, %ebx		F	D	E	W									
cmpl %ebx, %ecx			F	D	D	D	E	W						
movl ind, %eax				F	F	F	D	E	W					
jnz j1 #no							F	D	D	E	W			
subl %ebx, %eax								F	F	D	E	W		

Ottimizzando con branch prediction si ha che viene effettuata la fetch dell'istruzione all'etichetta j1, ma poi viene scartata perchè il salto non viene effettuato. Si ha quindi:

Istruzione	1	2	3	4	5	6	7	8	9	10	11	12	13	14
j1: addl %ebx, %ecx	F	D	E	W					F	F	F			
movl ind, %ebx		F	D	E	W									
cmpl %ebx, %ecx			F	D	D	D	E	W						
movl ind, %eax				F	F	F	D	E	W					
jnz j1 #no							F	D	D	E	W			
subl %ebx, %eax								F	F	D	E	W		