Esercizio 1:
Considerare le seguenti istruzioni agli indirizzi di memoria indicati:

```
1000  Load     R5, 200(R6)
1004  And      R2, R1, #0xF
1008  Add      R4, R3, R5
1012  Subtract R7, R4, R2
```

Inizialmente il registro R1 contiene 78, il registro R3 contiene 100, il registro R6 contiene 2000 e la locazione di memoria all'indirizzo 2200 contiene il valore 20. Assumere che le istruzioni vengano eseguite da un processore con Pipeline a 5 stadi.

1. Disegnare due diagrammi temporali che mostrano i vari stadi di esecuzione di ciascuna istruzione nei vari cicli di clock (assumere che il ciclo numero uno sia quello in cui avviene la fase di prelievo della prima istruzione). Si rappresenti nel primo diagramma il caso in cui il processore non esegua alcun inoltro di registri interstadio, mentre nel secondo il caso in cui vengano inoltrati sia il registro RZ che il registro RY
2. Mostrare una tabella con i contenuti dei registri IR, PC, RA, RB, RZ e RY per i cicli di clock dal 2 all'8 per il caso con inoltro di RZ e RY

---

1

Senza inoltri:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Load | P | D | E | M | S | | | | | | | | |
| And | | P | D | E | M | S | | | | | | | |
| Add | | | P | | | D | E | M | S | | | | |
| Subtract | | | | P | | | | | | D | E | M | S |

Con inoltri di RZ e RY:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Load | P | D | E | M | S | | | | | | | | |
| And | | P | D | E | M | S | | | | | | | |
| Add | | | P | D | E | M | S | | | | | | |
| Subtract | | | | P | D | E | M | S | | | | | |

2

| | IR | PC | RA | RB | RZ | RY |
|---|---|---|---|---|---|---|
| Ciclo 2 | [1000] | 1004 | * | * | * | * |
| Ciclo 3 | [1004] | 1008 | 2000 | [R5] | * | * |
| Ciclo 4 | [1008] | 1012 | 78 | [R2] | 2200 | * |
| Ciclo 5 | [1012] | 1016 | 100 | [R5] | 14 | 20 |
| Ciclo 6 | [1016] | 1020 | [R4] | [R2] | 120 | 14 |
| Ciclo 7 | [1020] | 1024 | * | * | 106 | 120 |
| Ciclo 8 | [1024] | 1028 | * | * | * | 106 |

Esercizio 2:
Si assuma che il 30% del conteggio dinamico delle istruzioni eseguite per un programma siano istruzioni di salto. Si assuma inoltre che il 60% dei salti siano realmente effettuati. Il programma viene eseguito in due processori diversi con la stessa frequenza di clock R. Uno usa la predizione statica dei salti con l'assunto di salto non effettuato. L'altro usa la predizione dinamica di salti.

1. Assumendo che non ci siano altri tipi di conflitti, quale deve essere la minima accuratezza di predizione per il processore che usa la predizione dinamica per avere almeno le stesse prestazioni del processore che usa la predizione di salto statica?
2. Se l'effettiva accuratezza di predizione dinamica risulta essere del 95%, qual e' lo speedup del throughput del processore con predizione dinamica rispetto a quello con predizione statica? Sara' piu' lento o piu' veloce?

---

1

Per valutare le prestazioni dei due processore possiamo calcolare il loro throughput $P = R/(S + \delta)$, dove S e' il numero medio di cicli per istruzione e $\delta$ e' il numero di stalli medio introdotto dai salti.

Per il processore con predizione statica il $\delta$ sara':

$\delta_{statica}$ = Conteggio istruzioni di salto · numero di salti effettuati · cicli di stallo in caso di predizione errata = $0.3 \cdot 0.6 \cdot 1 = 0.18$

Per il processore con predizione dinamica il $\delta$ sara':

$\delta_{dinamica}$ = Conteggio istruzioni di salto · percentuale errori di predizione · cicli di stallo in caso di predizione errata = $0.3 \cdot x \cdot 1$

Il processore con predizione dinamica avra' prestazioni migliori o uguali di quello a predizione statica se:

$P_{dinamica} >= P_{statica} \rightarrow \delta_{dinamica} <= \delta_{statica} \rightarrow 0.3 \cdot x \cdot 1 <= 0.3 \cdot 0.6 \cdot 1 \rightarrow x <= 0.6$

Quindi il processore con predizione dinamica dovra' avere un'accuratezza di predizione almeno del 40%

2

Per il processore con predizione dinamica corretta il 95% delle volte il $\delta$ sara':

$\delta_{dinamica95}$ = Conteggio istruzioni di salto · percentuale errori di predizione · cicli di stallo in caso di predizione errata = $0.3 \cdot 0.05 \cdot 1 = 0.015$

Il calcolo dello speedup (quante volte e' piu' veloce) del processore con predizione dinamica rispetto a quello con predizione statica si ottiene dividendo i due throughput. Assumendo S = 1 avremo:

$P_{dinamica95} / P_{statica} = (R / (S + \delta_{dinamica95})) / (R / (S + \delta_{statica})) = (S + \delta_{statica}) / (S + \delta_{dinamica95}) = 1.18 / 1.015 = 1.16$

Quindi il processore a predizione dinamica avra' un incremento nel throughput di 1.16 volte. Se si volesse calcolare il valore in percentuale bastera' sottrarre 1 e moltiplicare per 100:

(1.16 – 1) * 100 = 16% di speedup

Esercizio 3:
Si consideri il seguente ciclo di istruzioni assembly:

```
CICLO   Load              R5, (R4)
        Add               R3, R3, R5
        Add               R4, R4, #4
        Subtract          R2, R2, #1
        Branch_if_[R2]>0  CICLO
        Store             R3, SOMMA
```

Si assuma sia eseguito da un processore con pipeline a 5 stadi e inoltro all'ALU dei registri RZ e RY. Disegnare un diagramma temporale per l'esecuzione di due iterazioni successive del ciclo per i seguenti casi:

a) Si assuma la pipeline usi la predizione statica di salto assumendo che il salto non venga mai eseguito
b) Stesse assunzioni del caso a, ma riordinando prima le istruzioni per ottimizzare le prestazioni
c) Si assuma che la pipeline usi salto differito con un posto di ritardo. Riordinare le istruzioni come necessario per migliorare le prestazioni

---

a

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Load | P | D | E | M | S | | | | | | | | | | | | |
| Add1 | | P | | D | E | M | S | | | | | | | | | | |
| Add2 | | | P | | D | E | M | S | | | | | | | | | |
| Subtract | | | | P | | D | E | M | S | | | | | | | | |
| Branch | | | | | P | | | D | E | M | S | | | | | | |
| Store | | | | | | P | - | - | - | - | - | - | - | - | - | - | - |
| Load | | | | | | | P | | D | E | M | S | | | | | |
| Add1 | | | | | | | | P | | | D | E | M | S | | | |
| Add2 | | | | | | | | | P | | | D | E | M | S | | |
| Subtract | | | | | | | | | | P | | | D | E | M | S | |
| Branch | | | | | | | | | | | P | | | | D | E | M |
| Store | | | | | | | | | | | | P | | | | D | E |

In questo caso si perdono 2 cicli di clock per iterazione.

b

Per ottimizzare si puo' invertire l'ordine delle due add:

```
CICLO   Load              R5, (R4)
        Add               R4, R4, #4        ; Add2
        Add               R3, R3, R5        ; Add1
        Subtract          R2, R2, #1
        Branch_if_[R2]>0  CICLO
        Store             R3, SOMMA
```

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Load | P | D | E | M | S | | | | | | | | | | | | |
| Add2 | | P | D | E | M | S | | | | | | | | | | | |
| Add1 | | | P | D | E | M | S | | | | | | | | | | |
| Subtract | | | | P | D | E | M | S | | | | | | | | | |
| Branch | | | | | P | | D | E | M | S | | | | | | | |
| Store | | | | | | P | - | - | - | - | - | - | - | - | - | - | - |
| Load | | | | | | | P | D | E | M | S | | | | | | |
| Add2 | | | | | | | | P | D | E | M | S | | | | | |
| Add1 | | | | | | | | | P | D | E | M | S | | | | |
| Subtract | | | | | | | | | | P | D | E | M | S | | | |
| Branch | | | | | | | | | | | P | | D | E | M | S | |
| Store | | | | | | | | | | | | P | | D | E | M | S |

In questo caso si perde un ciclo di clock per iterazione.

c

Per ottimizzare si puo' invertire l'ordine delle due add:

```
CICLO       Load                R5, (R4)
            Subtract            R2, R2, #1
            Add                 R3, R3, R5          ; Add1
            Branch_if_[R2]>0    CICLO
            Add                 R4, R4, #4          ; Add2
            Store               R3, SOMMA
```

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Load | P | D | E | M | S | | | | | | | | | | | | |
| Subtract | | P | D | E | M | S | | | | | | | | | | | |
| Add1 | | | P | D | E | M | S | | | | | | | | | | |
| Branch | | | | P | D | E | M | S | | | | | | | | | |
| Add2 | | | | | P | D | E | M | S | | | | | | | | |
| Load | | | | | | P | D | E | M | S | | | | | | | |
| Subtract | | | | | | | P | D | E | M | S | | | | | | |
| Add1 | | | | | | | | P | D | E | M | S | | | | | |
| Branch | | | | | | | | | P | D | E | M | S | | | | |
| Add2 | | | | | | | | | | P | D | E | M | S | | | |
| Store | | | | | | | | | | | P | | D | E | M | S | |

In questo caso non si perde alcun ciclo di clock per iterazione, ma solo due al termine

Esercizio 4:
Si considerino 2 processori visti a lezione aventi la stessa frequenza di clock: il processore A con pipeline a 5 stadi semplice; il processore B superscalare con 2 unita' di esecuzione (una aritmetica e l'altra load/store). Assumere che non ci siano dipendenze di dato, che non avvengano cache miss e che siano presenti due chace separate per istruzioni e dati. Si assuma inoltre un'accuratezza di predizione dei salti del 100%. Calcolare il miglior tempo di esecuzione di entrambi i processori e confrontarli tra di loro per l'esecuzione dei seguenti programmi:

a) Un programma senza istruzioni di salto dove il 70% delle istruzioni sono aritmetiche e il restante 30% sono istruzioni di accesso alla memoria
b) Un programma dove il 10% delle istruzioni sono di salto (con salti mai effettuati), il 55% sono aritmetiche e il 35% sono di accesso alla memoria

---

Il tempo di esecuzione T e' uguale a:

$T = N * S / R$

Con N numero di istruzioni, S cicli per istruzione e R frequenza del processore in cicli di clock al secondo.

a

Nel caso del processore A:

$T_A = N * 1 / R$

Nel caso del processore B le prestazioni migliori si ottengono alternando il piu' possibile istruzioni di accesso alla memoria con istruzioni aritmetiche. Avendo in media piu' istruzioni aritmetiche, sara' il numero di queste ultime a determinare quanti cicli di clock sono necessari in media per ogni istruzione. Dato che si hanno 70% di istruzioni aritmetiche e che il restante 30% di istruzioni di accesso alla memoria possono essere eseguite in parallelo con queste ultime, il numero medio di cicli per istruzione e' pari a 0.7. Quindi il tempo di esecuzione sara':

$T_B = N * 0.7 / R$

Calcoliamo ora quante volte $T_A$ e' piu' lento di $T_B$:

$T_A / T_B = 1 / 0.7 = 1.43$

In percentuale:

$(1.43 – 1) * 100 = 43\%$

b

Avendo un'accuratezza del 100% sulle predizioni dei salti, il processore A continuera' ad avere S = 1. Quindi il suo tempo di esecuzione sara':

$T_A = N * 1 / R$

Nel caso del processore B le istruzioni di salto vengono predette nella fase di prelievo, quindi non passano mai attraverso le unita' di esecuzione e, avendo precisione di predizione del 100%, non influenzano il tempo di esecuzione. Per le istruzioni aritmetiche e di accesso alla memoria vale il ragionamento fatto per il punto a. Quindi  il numero medio di cicli per istruzione e' pari a 0.55

$T_B = N * 0.55 / R$

Calcoliamo ora quante volte $T_A$ e' piu' lento di $T_B$:

$T_A / T_B = 1 / 0.55 = 1.82$

In percentuale:

$(1.82 – 1) * 100 = 82\%$

Esercizio 5
Si assuma che il 25% del conteggio dinamico delle istruzioni eseguite per un programma siano istruzioni di salto. Si usi il salto differito, con un posto del ritardo. Si assuma che non ci siano stalli causati da altri fattori. Determinare un'espressione per il tempo di esecuzione espresso in cicli per i seguenti casi:

a) Qualora tutti i posti del ritardo siano occupati con istruzioni NOP.
b) Qualora il 60% di posti di ritardo occupati con istruzioni utili dal compilatore ottimizzante. Determinare l'incremento delle prestazioni, espresso come speedup del throughput, del caso b rispetto al caso a
c) Usando il salto differito con 2 posti del ritardo, qualora il primo posto venga occupato con un'istruzione utile il 75% delle volte, ma entrambi i posti di ritardo vengano occupati da istruzioni utili solo il 15% delle volte

---

Il tempo di esecuzione T e' uguale a:

$T = N * S / R$

Con N numero di istruzioni, S cicli per istruzione e R frequenza del processore in cicli di clock al secondo.

a

Il salto differito esegue in ogni caso l'istruzione dopo il salto. Nel caso tutti i posti di ritardo siano rimpiazzati da NOP avremo un ciclo di clock di ritardo per ogni istruzione di salto. Quindi:

$S = 1 + 0.25 * 1 = 1.25$

$T = N * 1.25 / R$

b

Se il 60% di posti di ritardo contiene istruzioni utili allora avremo 1 ciclo di clock di ritardo per il restante 40% delle istruzioni di salto. Quindi:

$S = 1 + 0.25 * 0.4 * 1 = 1.1$

$T = N * 1.1 / R$

L'incremento di prestazioni del caso b rispetto al caso a e' il seguente

$Ta / Tb = 1.25 / 1.1 = 1.14$

in percentuale:

$(1.14 – 1) * 100 = 14\%$

Nel caso b l'ottimizzazione effettuata dal compilatore ha portato ad un incremento delle prestazioni del 14%

c

In questo caso si hanno 2 posti di ritardo. Sappiamo che il 15% dei salti ha entrambi i posti di ritardo contenenti istruzioni utili. Per il 25% dei casi il compilatore non e' riuscito a trovare istruzioni utili da rimpiazzare, quindi avremo 2 cicli di clock di ritardo. Il restante 60% dei salti aggiungera 1 ciclo di clock di ritardo per istruzione, dato che  hanno solo un posto di ritardo contenente un'istruzione utile. Quindi avremo:

$S = 1 + 0.25 (0.25 * 2 + 0.6 * 1) = 1.28$

$T = N * 1.28 / R$