Esercizio 1:
mostrare il contenuto di tutti i registri interstadio (RA, RB, RZ, RM, PC_Temp, RY) e del registro PC per ognuno dei 5 stadi di prelievo ed esecuzione di un'istruzione. Usare l'architettura RISC vista a lezione. Assumere che l'indirizzo puntato da PC inizialmente sia 0xC36B0

1. Subtract R2, R3, #10 (R3 contiene il valore 20)
2. And R2, R2, R3 (R2 contiene 0xB430A e R3 contiene 0x200)
3. Load R2, 1000(R3) (R3 contiene 3000 e la locazione di memoria da leggere contiene il valore 5)
4. Call_Register R2 (R2 contiene 0x1000)

1

	RA	RB	RZ	RM	PC_Temp	RY	PC
Stadio 1	*	*	*	*	0xC36B0	*	0xC36B0
Stadio 2	*	*	*	*	0xC36B0	*	0xC36B4
Stadio 3	20	*	*	*	0xC36B4	*	0xC36B4
Stadio 4	20	*	10	*	0xC36B4	*	0xC36B4
Stadio 5	20	*	10	*	0xC36B4	10	0xC36B4

2

	RA	RB	RZ	RM	PC_Temp	RY	PC
Stadio 1	*	*	*	*	0xC36B0	*	0xC36B0
Stadio 2	*	*	*	*	0xC36B0	*	0xC36B4
Stadio 3	0xB430A	0x200	*	*	0xC36B4	*	0xC36B4
Stadio 4	0xB430A	0x200	0x200	0x200	0xC36B4	*	0xC36B4
Stadio 5	0xB430A	0x200	0x200	0x200	0xC36B4	0x200	0xC36B4

3

	RA	RB	RZ	RM	PC_Temp	RY	PC
Stadio 1	*	*	*	*	0xC36B0	*	0xC36B0
Stadio 2	*	*	*	*	0xC36B0	*	0xC36B4
Stadio 3	3000	1000	*	*	0xC36B4	*	0xC36B4
Stadio 4	3000	1000	4000	1000	0xC36B4	*	0xC36B4
Stadio 5	3000	1000	4000	1000	0xC36B4	5	0xC36B4

4

	RA	RB	RZ	RM	PC_Temp	RY	PC
Stadio 1	*	*	*	*	0xC36B0	*	0xC36B0
Stadio 2	*	*	*	*	0xC36B0	*	0xC36B4
Stadio 3	0x1000	*	*	*	0xC36B4	*	0xC36B4
Stadio 4	0x1000	*	*	*	0xC36B4	*	0x1000
Stadio 5	0x1000	*	*	*	0x1000	0xC36B4	0x1000

Esercizio 2:
Stilare la sequenza di passi necessari per prelevare ed eseguire in un architettura CISC a 3 BUS le seguenti istruzioni:

1. Move -(R2), R3
2. Or R2, 200(R3) (Assumere che il valore immediato sia contenuto nella parola di memoria successiva a quella dell'istruzione)
3. Move R2, (R3)+

---

1 Move -(R2), R3

1. Indirizzo di memoria ← [PC], Leggi memoria, Attesa MFC, IR ← Dati di memoria, PC ← [PC] + 4
2. Decodifica istruzione
3. R2 ← [R2] – 4
4. Indirizzo di memoria ← [R2], Dati ← [R3], Scrivi memoria, Attesa MFC

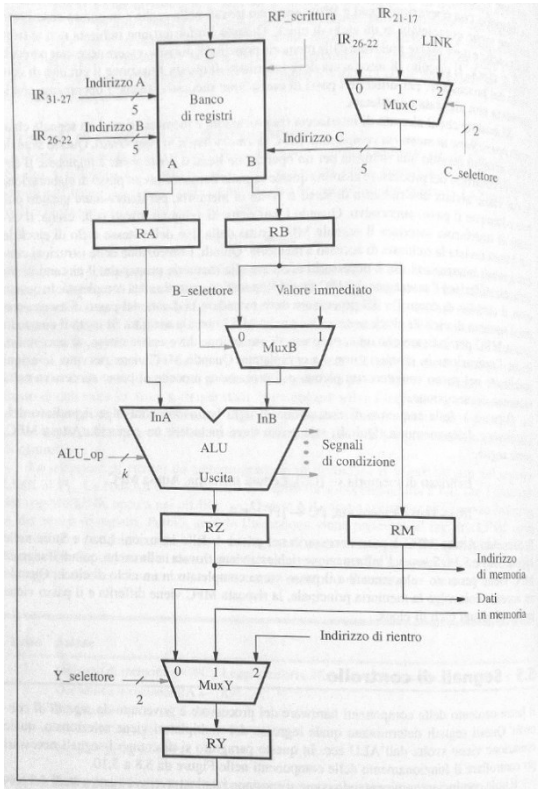
2 Or R2, 200(R3)

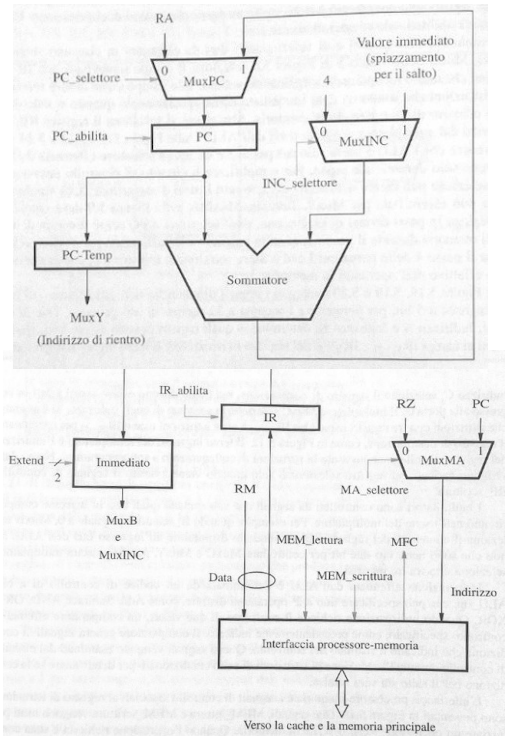
1. Indirizzo di memoria ← [PC], Leggi memoria, Attesa MFC, IR ← Dati di memoria, PC ← [PC] + 4
2. Decodifica istruzione
3. Indirizzo di memoria ← [PC], Leggi memoria, Attesa MFC, Temp1 ← Dati di memoria, PC ← [PC] + 4
4. Temp1← [R3] + [Temp1]
5. Indirizzo di memoria ← [Temp1], Leggi memoria, Attesa MFC, Temp1 ← Dati di memoria
6. R2 ← [R2] OR [Temp1]

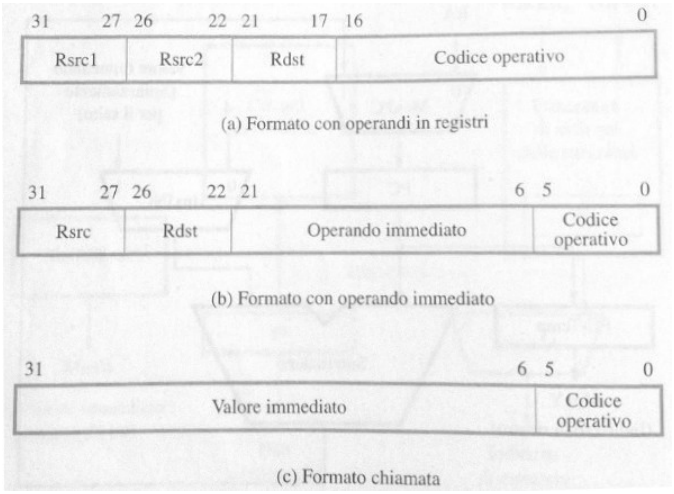
3 Move R2, (R3)+

1. Indirizzo di memoria ← [PC], Leggi memoria, Attesa MFC, IR ← Dati di memoria, PC ← [PC] + 4
2. Decodifica istruzione
3. Indirizzo di memoria ← [R3], Leggi memoria, Attesa MFC, R2 ← Dati di memoria
4. R3 ← [R3] + 4

Esercizio 3:
Derivare le espressioni logiche per generare i segnali C_selettore, MA_selettore, Y_selettore, PC_selettore e INC_selettore. Definire le variabili logiche utilizzate (ad esempio se si usasse una variabile chiamata BRANCH che vale 1 se viene eseguita un'istruzione di salto e 0 altrimenti).







Definiamo le variabili:

- 3Reg: che vale 1 per tutte le istruzioni con 3 registri come operandi
- Call: che vale 1 per le istruzioni di chiamata a sottoprogramma
- SottoProg: che vale 1 per le istruzioni di chiamata tramite registro e rientro da sottoprogramma

I due bit di selezione $C\_selettore_1$ e $C\_selettore_0$ devono essere fissati a 01 per le istruzioni a tre registri, 10 per le istruzioni di chiamata a sottoprogramma, e negli altri casi 00. Questo porta a:

$$C\_selettore_0 = 3Reg \qquad C\_selettore_1 = Call$$

MA_selettore = 0 seleziona il registro Z come sorgente dell'indirizzo di memoria, ma soltanto durante il passo 4 (T4). Seleziona il PC in altri momenti.

$$MA\_selettore = \neg(T4)$$

Y_selettore deve essere pari a 01 per le istruzioni Load, 10 per le istruzioni di chiamata a sottoprogramma, e 0 altrimenti.

$$Y\_selettore_0 = Load \qquad Y\_selettore_1 = Call$$

MuxPC seleziona RA al passo 3 (T3) delle istruzioni di chiamata (tramite registro) e rientro da sottoprogramma, negli altri casi seleziona l'uscita del sommatore. MuxINC seleziona il valore immediato al passo 3, per il suo uso in istruzioni di salto.

$$PC\_selettore = \neg(T3 \cdot SottoProg)$$
$$INC\_selettore = T3$$

Esercizio 4:
Un processore con controllo microprogrammato ha i seguenti ritardi:

1. Generazione dell'indirizzo di inizio della microutine: 2,4 ns
2. Lettura di una microistruzione dalla memoria di microprogramma: 1,8 ns
3. Operazione tramite ALU: 2,6 ns
4. Accesso alla memoria cache: 1,3 ns

Assumendo che tutte le istruzioni e i dati siano già presenti nella cache:

a) Determinare il tempo minimo necessario per ciascuno dei passi delle operazioni dell'esercizio 2

b) Supponendo non esistano altri ritardi, qual'è il tempo minimo assegnabile ad un ciclo di clock?

1

Move -(R2), R3

1. Lettura microistruzione + Accesso memoria = 1,8 ns + 1,3 ns = 3,1 ns
2. Lettura microistruzione + Generazione dell'indirizzo = 1,8 ns + 2,4 ns = 4,2 ns
3. Lettura microistruzione + ALU = 1,8 ns + 2,6 ns = 4,4 ns
4. Lettura microistruzione + Accesso memoria = 1,8 ns + 1,3 ns = 3,1 ns

Or R2, 200(R3)

1. Lettura microistruzione + Accesso memoria = 1,8 ns + 1,3 ns = 3,1 ns
2. Lettura microistruzione + Generazione dell'indirizzo = 1,8 ns + 2,4 ns = 4,2 ns
3. Lettura microistruzione + Accesso memoria = 1,8 ns + 1,3 ns = 3,1 ns
4. Lettura microistruzione + ALU = 1,8 ns + 2,6 ns = 4,4 ns
5. Lettura microistruzione + Accesso memoria = 1,8 ns + 1,3 ns = 3,1 ns
6. Lettura microistruzione + ALU = 1,8 ns + 2,6 ns = 4,4 ns

Move R2, (R3)+

1. Lettura microistruzione + Accesso memoria = 1,8 ns + 1,3 ns = 3,1 ns
2. Lettura microistruzione + Generazione dell'indirizzo = 1,8 ns + 2,4 ns = 4,2 ns
3. Lettura microistruzione + Accesso memoria = 1,8 ns + 1,3 ns = 3,1 ns
4. Lettura microistruzione + ALU = 1,8 ns + 2,6 ns = 4,4 ns

2

Il ciclo di clock potrà durare al minimo il tempo dell'istruzione più lenta. Quindi sarà 4,4 ns