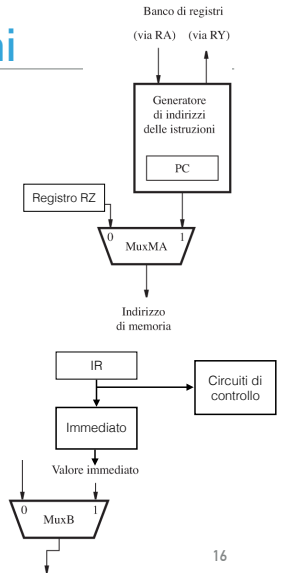
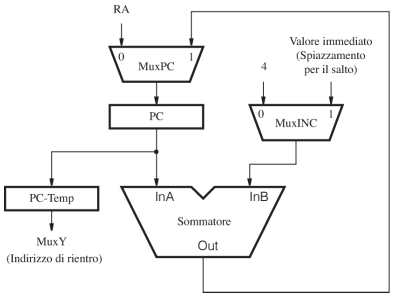
Sezione di prelievo delle istruzioni

- Si prelevano dalla memoria
  - un'istruzione, dalla locazione puntata da PC
  - un operando dalla locazione puntata da RZ, quest'ultima è calcolata con valori di altri operandi, es. con indirizzamento indiretto, `Load R5, X(R7)`
- MuxMA seleziona una sorgente e la invia alla memoria
- Il contenuto di IR va ai circuiti di controllo per generare i segnali di controllo
- IR può contenere un valore immediato a 16 bit che indica un indirizzo. Il componente Immediato riceve IR estrae l'indirizzo e lo estende a 32 bit, quindi lo inoltra a MuxB per accedere alla memoria



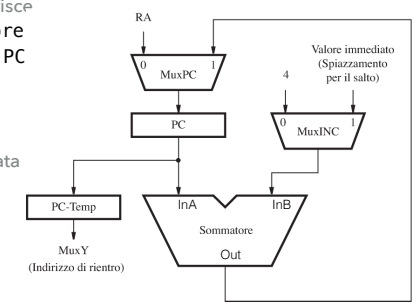
Generatore di indirizzi delle istruzioni

- Nell'esecuzione sequenziale il Sommatore prende in ingresso PC e 4 ed effettua la somma, così PC punta all'istruzione successiva nella sequenza
- Primo passo (fronte del clock)
  - L'istruzione da eseguire è prelevata dalla memoria e scritta in IR
  - PC è scritto in PC–Temp e al Sommatore si passa PC in InA, e 4 in InB
  - Il risultato della somma, ovvero Out, è scritto in PC e in PC–Temp (MuxPC ha selezionato l'ingresso 1). Quindi in PC e PC–Temp è stato scritto l'indirizzo dell'istruzione successiva (sequenziale)
- Secondo passo
  - L'istruzione in IR è decodificata



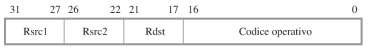
Generatore di indirizzi delle istruzioni

- Per istruzioni di salto, il Sommatore prende in ingresso PC e lo spiazzamento relativo a PC (operando immediato in IR)
- Terzo passo: si preleva lo spiazzamento da IR e si inserisce in InB (MuxINC ha selezionato l'ingresso 1), il Sommatore lo somma con PC che proviene da InA, si scrive Out su PC (MuxPC ha selezionato l'ingresso 1)
- Un'istruzione di chiamata a subroutine è stata assemblata come `Call_Register R9`, con R9 pari all'indirizzo della prima istruzione della subroutine
- Al passo 2, R9 era stato prelevato e inserito in RA
- Terzo passo: RA è scritto in PC (MuxPC ha selezionato l'ingresso 0)
- Al passo 2, in PC–Temp era stato inserito l'indirizzo di rientro da sottoprogramma o da interruzione



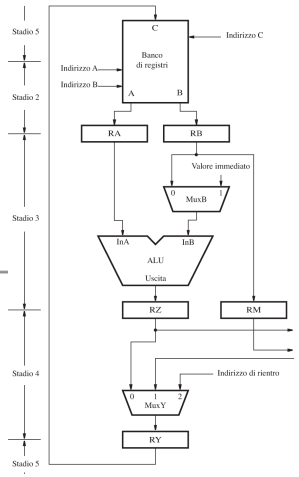
Passi di esecuzione (S. 5.4): Add

- L'istruzione Add è presente in IR e codificata come in figura



- Gli indirizzi dei registri sorgente sono disponibili nei campi IR_{31-27} e IR_{26-22}
- Questi campi sono collegati agli ingressi Indirizzo A e Indirizzo B del banco dei registri
- Il campo IR_{21-16} è usato per l'ingresso Indirizzo C
- Ogni passo da 2 a 5 è svolto dal corrispondente stadio del percorso dati

Passo	Azione	`Add R3, R4, R5`
1	Indirizzo di memoria ← [PC], Leggi memoria, IR ← Dati da memoria, PC ← [PC] + 4	
2	Decodifica istruzione, RA ← [R4], RB ← [R5]	
3	RZ ← [RA] + [RB]	
4	RY ← [RZ]	
5	R3 ← [RY]	



Passi di esecuzione: Load e Store

- Le istruzioni `Load` e `Store` sono codificate come in figura

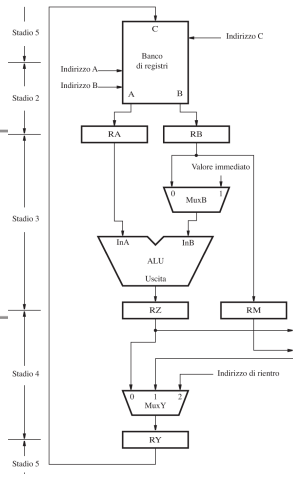
- Per `Load`, l'indirizzo del registro destinazione è in IR_{26-22} e collegato all'ingresso `Indirizzo C`

Passo **Azione** `Load R5, X(R7)`

1	Indirizzo di memoria ← [PC], Leggi memoria, IR ← Dati da memoria, PC ← [PC] + 4
2	Decodifica istruzione, RA ← [R7]
3	RZ ← [RA] + Valore immediato X
4	Indirizzo di memoria ← [RZ], Leggi memoria, RY ← Dati da memoria
5	R5 ← [RY]

Passo **Azione** `Store R6, X(R8)`

1	Indirizzo di memoria ← [PC], Leggi memoria, IR ← Dati da memoria, PC ← [PC] + 4
2	Decodifica istruzione, RA ← [R8], RB ← R6
3	RZ ← [RA] + Valore immediato X, RM ← [RB]
4	Indirizzo di memoria ← [RZ], Dati per memoria ← [RM], Scrivi in memoria
5	Nessuna azione



Passi di esecuzione: salti

- Formato delle istruzioni di salto

- Salto incondizionato

Passo	Azione
1	Indirizzo di memoria ← [PC], Leggi memoria, IR ← Dati da memoria, PC ← [PC] + 4
2	Decodifica istruzione
3	PC ← [PC] + Spiazzamento per il salto
4	Nessuna azione
5	Nessuna azione

- Salto condizionato, istruzione `Branch_if_[R5]=[R6] CICLO`
  - Il confronto può essere effettuato tramite sottrazione `[R5]-[R6]` nell'`ALU`, ma il circuito di confronto è più veloce, quindi il circuito di controllo esamina il risultato, ovvero positivo, negativo nullo, in base ai segnali generati

Passo	Azione
1	Indirizzo di memoria ← [PC], Leggi memoria, IR ← Dati da memoria, PC ← [PC] + 4
2	Decodifica istruzione, RA ← [R5], RB ← [R6]
3	Confronta [RA] con [RB], Se [RA] = [RB], allora PC ← [PC] + Spiazzamento per il salto
4	Nessuna azione
5	Nessuna azione

In attesa della memoria

- Si è assunto che le operazioni di caricamento dalla memoria e di scrittura in memoria siano completate in un ciclo di clock (nello stadio 4 per le istruzioni load e store)
- La memoria è **molto** più lenta del processore e richiede più di un ciclo di clock per essere letta
- Un processore 2 GHz ha un periodo di clock di 0,5 ns, la memoria RAM ha tempo di accesso di circa 100 ns. Lo stadio 4 dovrebbe durare 200 cicli di clock **:(**
- Tuttavia se il dato è nella cache L1 (sullo stesso chip del processore), il tempo di accesso è circa 0,5 ns, allora un ciclo di clock basta per la lettura. Il tempo di accesso alla cache L1 potrebbe essere fino a 2 ns (4 cicli)
- Se il dato non è in cache, il circuito di controllo estende il passo di esecuzione fino a quando l'operazione di lettura non sia stata completata
- Al completamento della lettura viene asserito il segnale `MFC` (memory function completed)

Lettura registri sorgente

- Si è assunto che i registri sorgenti siano letti al passo 2 contemporaneamente alla decodifica del codice operativo dell'istruzione appena caricata
- Poiché l'istruzione non è stata ancora decodificata come si determinano i registri da caricare?
- Gli indirizzi dei registri da caricare sono negli stessi bit per tutte le istruzioni, l'hardware li legge non appena l'istruzione è stata caricata in `IR` e sono disponibili in `RA` e `RB` alla fine del passo 2
- Se non sono necessari, al passo 3 essi saranno ignorati dall'hardware
- Per ciascuna istruzione vista prima, al passo 2 vengono letti due registri, quando non serve uno dei registri, questo viene ignorato

Segnali di controllo (S. 5.5)

- Durante l'esecuzione di un'istruzione i dati si spostano attraverso i quattro stadi nel percorso dati e i risultati delle azioni svolte sono immagazzinati nei registri interstadi `RA`, `RB`, `RZ`, `RY`, `RM`, `PC–Temp` e trasferiti da uno stadio al successivo in ciascun ciclo di clock, quindi **i registri interstadi sono sempre abilitati**
- I contenuti dei registri `PC`, `IR` non devono essere alterati a ogni ciclo di clock, quindi **sono abilitati solo nei passi in cui è necessaria una scrittura** (i segnali di controllo forniscono l'abilitazione alla scrittura)
- Nel **percorso dati**, i **segnali di controllo** forniscono l'input di selezione ai multiplatori `MuxB`, `MuxY` e `MuxC`. Occorre inviare la selezione al `MuxB` **al passo 3**, ma si invia la stessa selezione in tutti i passi per semplificare il circuito di controllo (lo stesso per `MuxY`)
- Nella sezione di prelievo, il segnale di controllo dà l'input per `MuxMA`, la selezione deve cambiare dal passo 1 (uso del `PC` per prelevare l'istruzione) al passo 4 (uso di `RZ` per `Load` e `Store`)
- Nel generatore di indirizzi, i segnali di controllo forniscono l'input per `MuxINC` e `MuxPC`
- Inoltre occorre inviare un segnale di scrittura al banco dei registri, selezionare l'operazione dell'`ALU`, selezionare il bit di esito, mandare il segnale di scrittura o lettura alla, e `MFC`