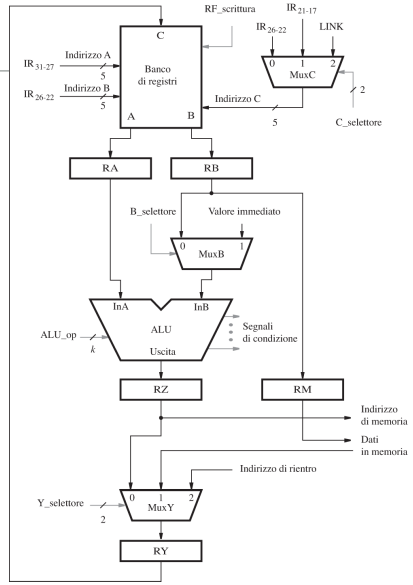
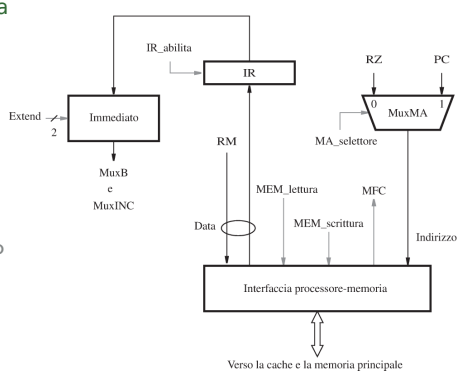
Segnali di controllo

- Occorre inviare 3 indirizzi a 5 bit al banco di memoria. Per `Indirizzo A` e `B` si collegano i campi appropriati di `IR`, per `Indirizzo C` il `C_selettore` di `MuxC` seleziona l'ingresso (per call subroutine l'ingresso è `Link`)
- `RF_scrittura` abilita la scrittura sul registro
- `B_selettore` è un bit di selezione per `MuxB`, mentre `C_selettore` e `Y_selettore` sono 2 bit, poiché si hanno 3 ingressi
- `ALU_op` è un segnale di controllo di k bit, per specificare 2^k operazioni dell'`ALU`



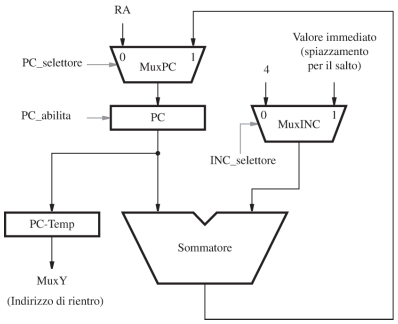
Segnali per l'interfaccia processore-memoria

- Due segnali `Mem_lettura` e `Mem_scrittura` permettono di iniziare operazioni corrispondenti a istruzioni `Load` e `Store`, ovvero read e write in memoria. Inoltre, `Mem_lettura` è usato per prelevare nuove istruzioni
- Per il registro `IR`, a ogni passo di prelievo si abilita la scrittura su `IR`, tramite `IR_abilita`, solo dopo che il segnale `MFC` sia stato asserito dall'interfaccia processore-memoria
- Per il blocco `Immediato` bisogna selezionare il tipo di estensione. Si hanno tre opzioni: valore a 16 bit con segno, 16 bit senza segno, 26 bit; quindi `Extend` è di 2 bit



Segnali al generatore di indirizzi

- `INC_selettore` seleziona il valore da aggiungere a `PC`
- `PC_selettore` seleziona `RA` o valore aggiornato del `PC`, che viene scritto su `PC` quando `PC_abilita` è attivato

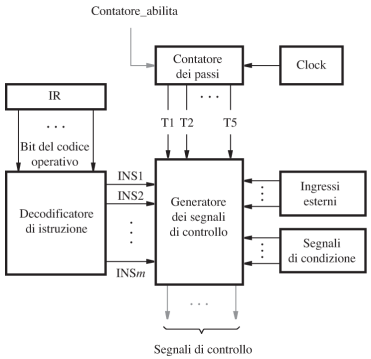


Controllo di tipo cablato (S. 5.6)

- Per generare le sequenze di segnali di controllo vi sono due metodologie: controllo cablato e microprogrammato
- Il controllo cablato (hardware) si realizza con una rete combinatoria che a ogni ciclo di clock genera i segnali di controllo in base ai seguenti ingressi
  - passo della sequenza (da 1 a 5)
  - istruzione corrente, ovvero contenuto del registro `IR`
  - bit di esito correnti (provenienti dall'`ALU`)
  - segnali provenienti dall'esterno, come le interruzioni

Generazione di segnali di controllo

- Il **Decodificatore** di istruzione interpreta il codice operativo in `IR` e pone a 1 una uscita INSi
- Il contatore di passi pone a 1 una uscita da T1 a T5 per indicare quale passo corrente è in corso (tutte le istruzioni si eseguono in 5 passi)
- Il **Generatore** (al centro) produce i segnali di controllo necessari
- Es. per il passo 1 (prelievo istruzione da memoria), identificato da T1 a 1, il Generatore pone le sue uscite
  - `MA_selettore` a 1 per selezionare il `PC` su `MuxMA`
  - `Mem_lettura` a 1 per effettuare la lettura da memoria
  - `IR_Abilita` a 1 (solo quando `MFC` è 1)
  - `Inc_selettore` a 0 e `PC_selettore` a 1 per incrementare `PC` di 4
  - `PC_abilita` a 1 per scrivere il risultato di `PC` alla fine del passo 1



Segnali di controllo del percorso dati

- Le impostazioni per i vari segnali di controllo possono essere determinate a partire dalle azioni svolte in ciascun passo di esecuzione per ciascuna istruzione
- Es. per la scrittura sul banco dei registri `RF_scrittura` è posto a 1 nel passo 5 solo per le istruzioni che scrivono dati su registri (`Load`, `Add`, etc.)
- `RF_scrittura` può essere generato da un'espressione logica
  `RF_scrittura = T5 · (ALU + Load + Call)`
- Ovvero, `RF_scrittura` vale 1 se si è al passo 5 (T5 = 1) **e** il codice operativo dell'istruzione è un'operazione Aritmetico-Logica **o** una Load **o** una Call (nella formula `ALU`, `Load` e `Call` indicano insiemi di istruzioni)
- Altri segnali di controllo, come `B_selettore` non dipendono dal passo di esecuzione, quindi si ha `B_selettore = Immediato`, ovvero `B_selettore` vale 1 per tutte le istruzioni che hanno un valore immediato nell'`IR`

Ritardo della memoria

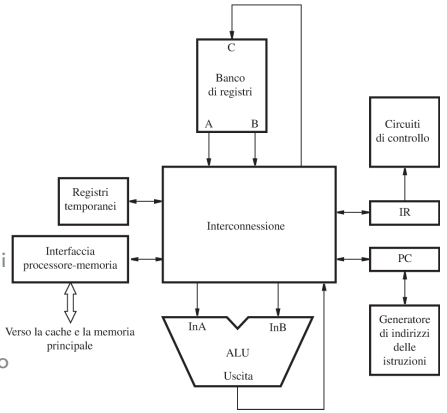
- Il contatore di passi viene incrementato alla fine di ciascun ciclo di clock
- Tuttavia, i passi in cui viene emesso un comando di lettura o scrittura dalla memoria non terminano fino a quando `MFC` non sia asserito
- Per estendere la durata del passo di esecuzione bisogna disabilitare il contatore di passi
- Sia `WMFC` un segnale che indica la necessità di aspettare un'operazione in memoria, allora `Contatore_abilita = not(WMFC) + MFC`
- Inoltre, la scrittura del `PC` avviene al passo 1 dopo l'arrivo di `MFC`, e al passo 3 solo per le istruzioni di salto
- Sia `BR` l'insieme di istruzioni che causano salti, si ha quindi
  `PC_abilita = T1 · MFC + T3 · BR`

Processori CISC (S. 5.7)

- I processori CISC hanno un'organizzazione hardware più complessa, poiché
  - grande varietà di codici operativi e modi di indirizzamento
  - operazioni con operandi in memoria, non solo Load/Store
  - lunghezza variabile delle istruzioni (non sempre una parola di memoria)

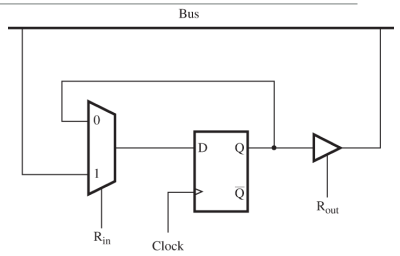
Organizzazione di un processore CISC

- La principale differenza con l'organizzazione a 5 stadi è il blocco **Interconnessione** che non prescrive un flusso di dati prefissato
- Il blocco **Interconnessione** permette di trasferire dati da qualunque coppia di componenti, per poter eseguire istruzioni che lavorano su dati in memoria
- Il banco **Registri temporanei** mantiene alcuni risultati intermedi durante l'esecuzione di istruzioni (usato al posto dei registri interstadi)
- In genere, il blocco **Interconnessione** è realizzato tramite bus. Un bus è un insieme di linee a cui sono connessi vari dispositivi
- Tramite i bus i dati possono essere trasferiti da qualunque dispositivo a qualsiasi altro



Controllo di accesso al bus

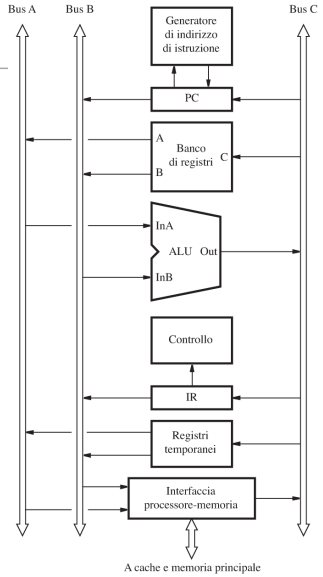
- Una porta logica che invia segnali sul bus viene chiamata **pilota del bus** (bus driver) e controlla l'accesso al bus tramite una porta a tre stati (tri-state)
- I vari dispositivi connessi al bus possono inviare dati, ma solo uno deve pilotare il bus in un dato momento
- La porta a tre stati ha un ingresso di controllo che quando è disabilitato disconnette elettricamente la porta dal bus
- Se `Rin` è 1, il multiplatore seleziona i dati dalla linea del bus e li invia al flip-flop. Se `Rin` è 0 il flip-flop mantiene il suo valore corrente
- L'uscita dal flip-flop è collegata al bus tramite una porta a tre stati, abilitata quando `Rout` è 1. Quando è abilitata essa può inviare un segnale logico sul bus, altrimenti è disabilitata



Interconnessione a tre bus

- Considerando `Add R5, R6`, ovvero `[R5] ← [R5] + [R6]`
- Prelievo e esecuzione di questa istruzione sono svolti in tre passi
- Passo 1: si invia il contenuto del `PC` all'interfaccia processore-memoria sul Bus B, e si inviano i dati dalla memoria a `IR` sul Bus C
- Passo 2: l'istruzione è decodificata e si leggono i registri `R5` e `R6`
- Passo 3: le uscite A e B del banco di registri sono disponibili e inviate all'ALU usando i Bus A e B. L'ALU effettua la somma e invia il risultato sul Bus C, quindi alla fine del ciclo di clock del passo 3, si scrive nel registro `R5`
- La lettura dei registri sorgenti inizia **dopo** che l'istruzione sia stata parzialmente decodificata poiché nei CISC non sempre lo stesso campo dell'istruzione codifica i registri

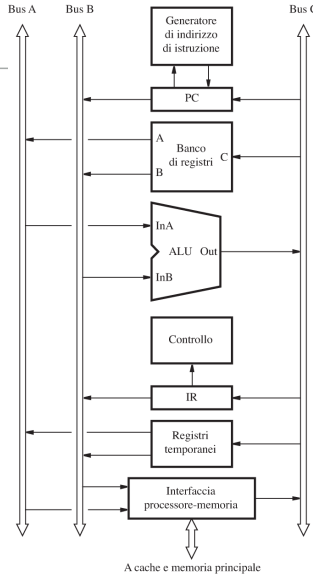
Passo	Azione
1	Indirizzo di memoria ← [PC], Leggi memoria, Attesa MFC, IR ← Dati da memoria, PC ← [PC] + 4
2	Decodifica istruzione
3	R5 ← [R5] + [R6]



Esempio con operando in memoria

- Consideriamo `And X(R7), R9`, ovvero AND logico sui contenuti del registro `R9` e la locazione di memoria `X+[R7]`
- Assumiamo che l'istruzione sia in due parole, la seconda contiene lo spiazzamento `X` a 32 bit. Si accede alla memoria 4 volte, i passi sono:

1. Lettura istruzione da memoria e incremento `PC`, ovvero: Indirizzo memoria ← `[PC]`, Leggi memoria, Attesa `MFC`, `IR` ← Dati da memoria, `PC←[PC]+4`
2. Decodifica istruzione
3. Lettura seconda parola dell'istruzione, ovvero: Indirizzo memoria← `[PC]`, Leggi memoria, Attesa `MFC`, Temp1←Dato X da memoria, `PC←[PC]+4`
4. Calcolo indirizzo `X(R7)`, ovvero: `Temp2←[Temp1]+[R7]`
5. Lettura dato in locazione `X(R7)`, ovvero: Indirizzo memoria ← [Temp2], Leggi memoria, Attesa `MFC`, Temp1 ← Dati da memoria
6. Calcolo AND, ovvero: Temp1 ← [Temp1] AND [R9]
7. Scrittura risultato in memoria, ovvero: Indirizzo memoria ← [Temp2], Dati memoria ← Temp1, Scrivi in memoria, Attesa `MFC`

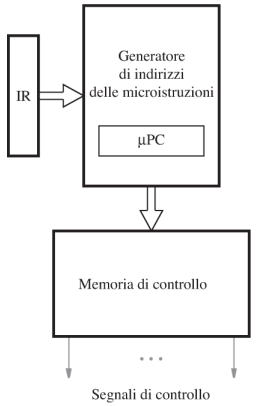


Controllo microprogrammato

- Non è detto che a sequenze di esecuzione più brevi di un processore CISC corrisponda minor tempo di esecuzione
- Concetti di base della microprogrammazione
  - **parola di controllo** (control word): insieme dei segnali di controllo in un certo istante di tempo. Ad ogni passo di esecuzione corrisponde una parola di controllo
  - **microistruzione**: parola di controllo impartita all'inizio di un passo di controllo
  - **microroutine**: sequenza temporale di microistruzioni di esecuzione di un'istruzione
- microistruzioni di prelievo e codifica comuni a tutte le microroutine
- microroutine specifica per un'istruzione inizia dal terzo passo
- microprogramma: interprete di istruzioni costituito dalle microroutine per tutte le istruzioni

Unità di controllo microprogrammato

- Memoria di controllo contiene le μ-routine
- Le μ-istruzioni sono indirizzate dal μPC, aggiornato dal generatore di μ-indirizzi
- Decodifica del codice operativo e la modalità di indirizzamento in [IR], quindi trova il μ-indirizzo della μ-istruzione iniziale della μ-routine che governa l'esecuzione dell'istruzione
- In ciascun passo la μ-routine, attraverso ciascuna μ -istruzione indica i segnali di controllo attivi



Controllo cablato-microprogrammato

- Il controllo μ-programmato è semplice da realizzare e fornisce flessibilità, tuttavia è più lento rispetto al controllo cablato
- I processori RISC non necessitano di tale flessibilità
- I segnali di controllo per realizzare i processori RISC sono generabili in modo relativamente semplice
- Il costo dei circuiti logici non è più un fattore significativo, quindi il controllo cablato è diventata la scelta preferita