Calcolatori Elettronici B
a.a. 2008/2009

Tecniche Pipeline: Elementi di base (ESERCIZI)

Massimiliano Giacomin

Esercizio – confronto prestazioni pipeline vs. multiciclo

- Si consideri la seguente combinazione di istruzioni eseguite
  - 22% load, 11% store, 49% formato-R, 16% salti condizionati, 2% salti incondizionati
- Si consideri la pipeline MIPS a 5 stadi con i seguenti tempi di esecuzione:
  - Prelievo, esecuzione con ALU, accesso in memoria = 4 ns
  - Decodifica/lettura registri, scrittura registri = 2 ns

Determinare il tempo medio di esecuzione per istruzione per un processore con controllo multiciclo e per un processore con pipeline, considerando per la pipeline il **caso ideale**: situazione ***a regime*** e ***nessuna criticità.*** Confrontare le prestazioni delle sue soluzioni.

Soluzione

T_{clock} = 4 ns

(sia nel caso multiciclo sia nel caso di pipeline:
corrisponde al tempo di esecuzione dell'unità funzionale più lenta)

Tempo medio di CPU nel caso di controllo multi-ciclo

- Numero di cicli per ciascuna classe di istruzioni
  - Load = 5
  - Store = 4
  - Formato-R = 4
  - Salti cond. = 3
  - Salti incond. = 3

- CPI = 0,22×5 + 0,11×4 + 0,49×4 + 0,16×3 + 0,02×3 = 4,04

- Tempo medio per istruzione = CPI × T_{clock}
  = 4,04 × 4 ns = 16,16 ns

Pipeline ideale e confronto prestazioni

- Per una pipeline ideale si considera CPI = 1
- Un ciclo di clock è pari a 4 ns
- Il tempo medio di esecuzione è quindi pari a

$$CPI * T_{clock} = 1 \times 4 \text{ ns}$$

- Confrontiamo ora i tempi medi nei due schemi considerati

$$\text{Speedup} = \frac{\text{Tempo}_{\text{multiciclo}}}{\text{Tempo}_{\text{pipeline}}} = \frac{16{,}16}{4} = 4{,}04$$

Le prestazioni migliorano di 4,04 volte nel caso di controllo pipeline

Appello 3 luglio 2006: ES. 1 [5 punti]

Si consideri, mostrato alla pagina seguente, il datapath corrispondente all'implementazione con tecnica pipeline a 5 stadi relativamente alle istruzioni MIPS *lw*, *sw*, *beq*, *Tipo-R*. Si vuole implementare la nuova istruzione

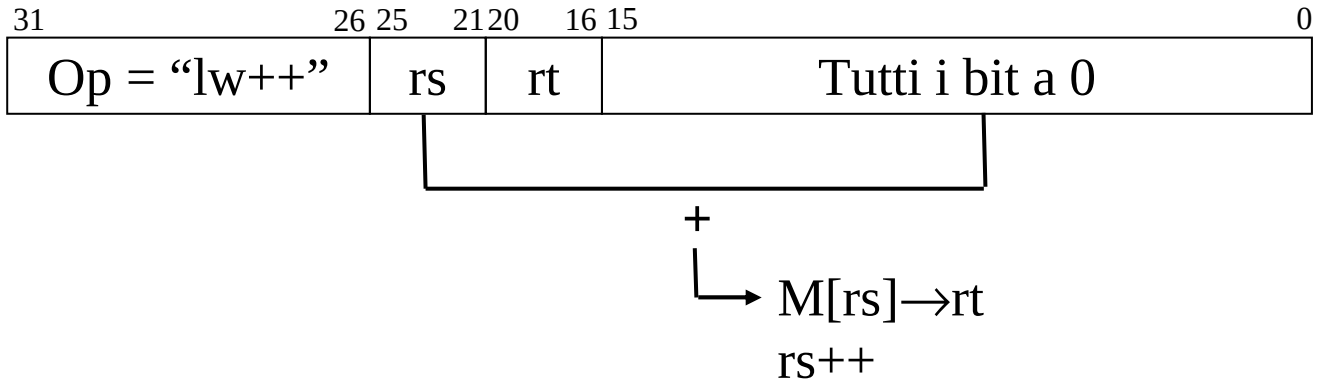
lw rt, (rs)++ rt←M[rs], rs←rs+4

che carica nel registro *rt* il valore della locazione di memoria di indirizzo specificato in *rs* ed incrementa il registro *rs* in modo che punti alla parola successiva in memoria. Si supponga in ogni caso che *rs* sia diverso da *rt*.

Ricordando i tre formati di codifica delle istruzioni (riportati di seguito) si chiede di:

- riportare il formato della nuova istruzione macchina
- riportare, nella corrispondente figura, le modifiche necessarie al datapath
- specificare come viene implementato il controllo della pipeline, riportando nel datapath l'unità di controllo e precisando (solo per la nuova istruzione) i segnali di controllo settati.

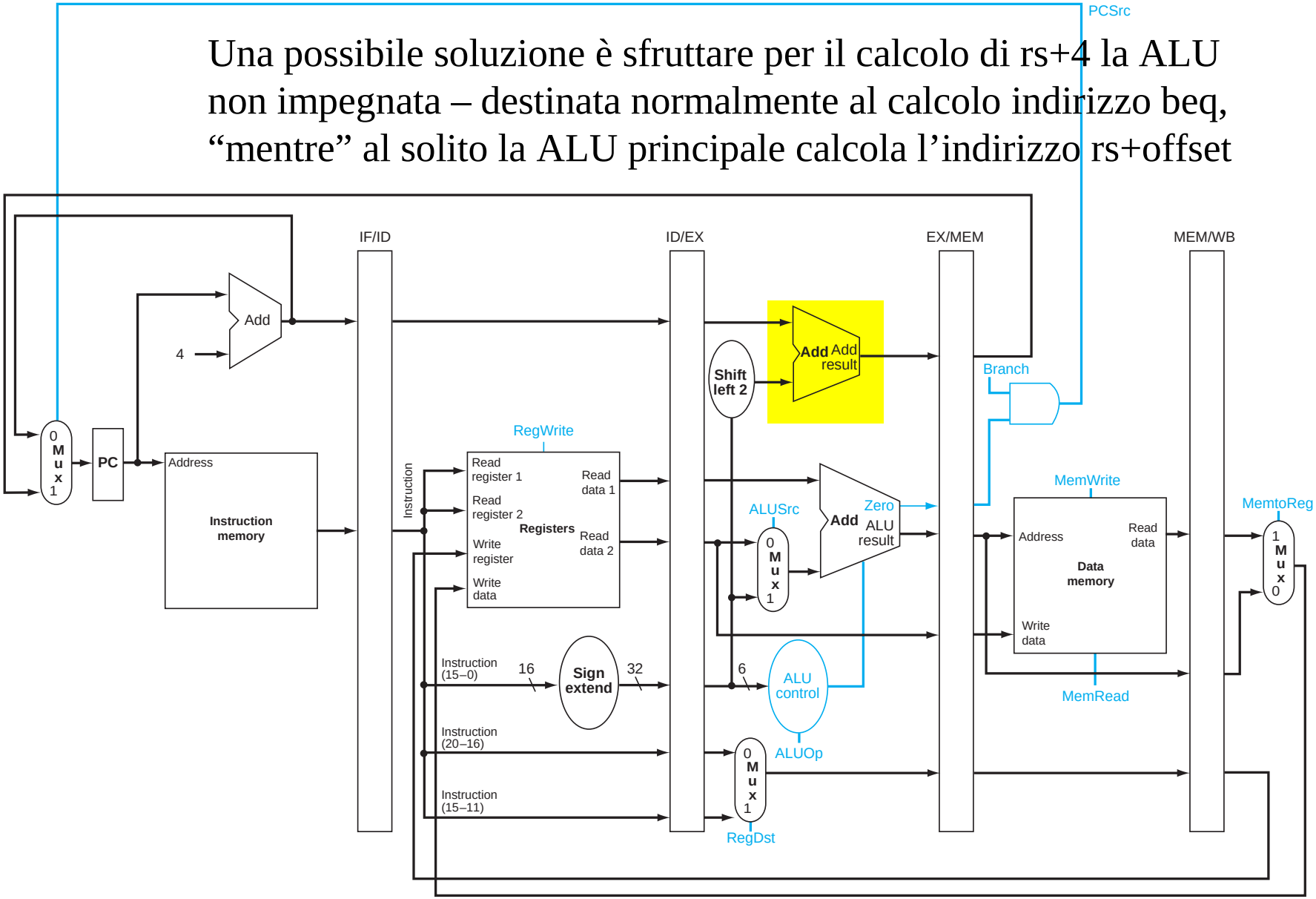
E’ possibile implementare l’istruzione con il **formato Tipo-I**:



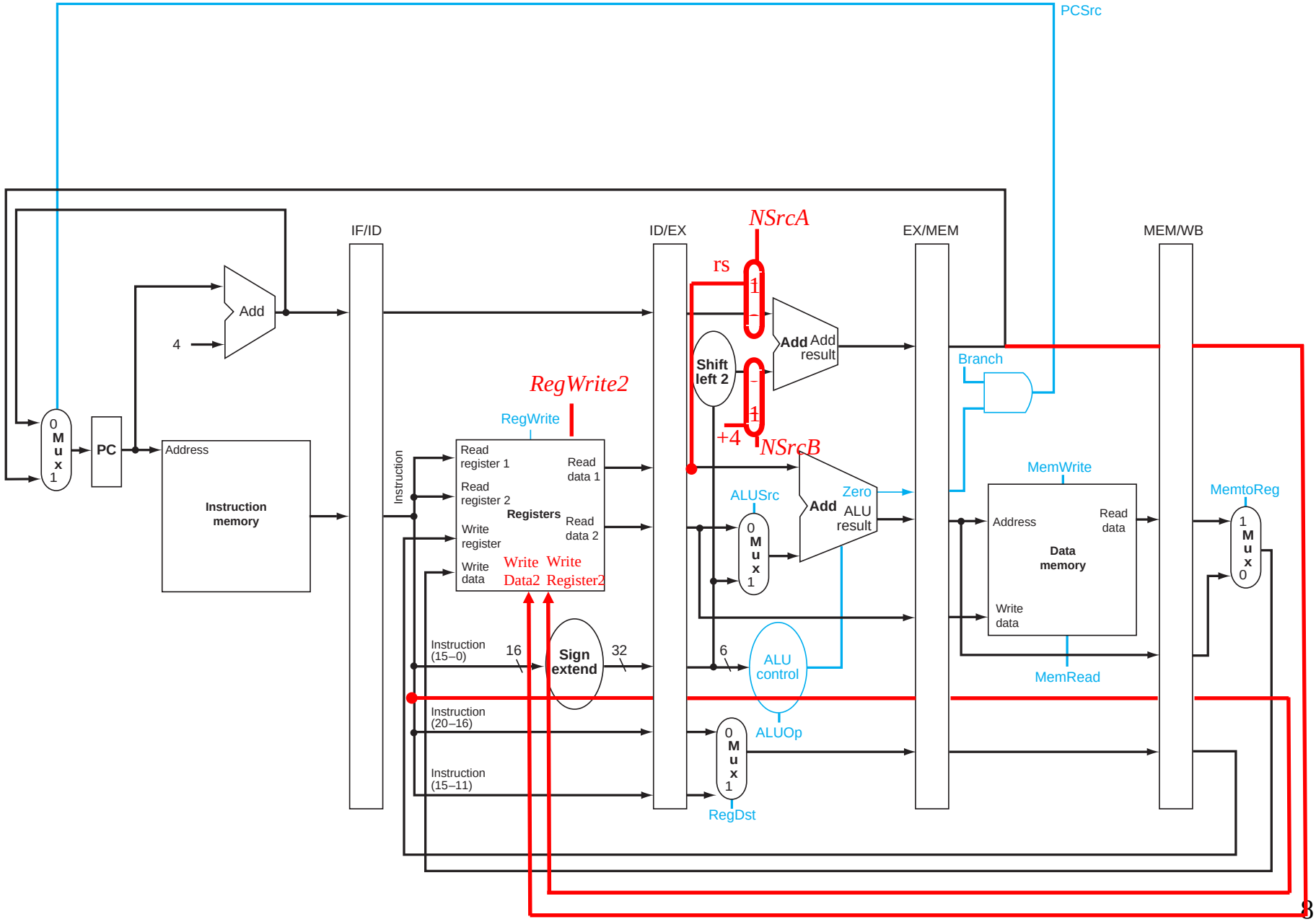
Vediamo il datapath…

Formato: TIPO-I (memorizza rs e rt) con offset a 0

Una possibile soluzione è sfruttare per il calcolo di rs+4 la ALU non impegnata – destinata normalmente al calcolo indirizzo beq, “mentre” al solito la ALU principale calcola l’indirizzo rs+offset



Dobbiamo aggiungere opportuni MUX alla ALU e segnali di controllo al Register file per poter scrivere due registri (rt e rs)



Controllo: combinatorio, nello stadio D.

Nel datapath, indicare l'unità di controllo come in Fig. del Patterson

Segnali di controllo per la nuova istruzione:

NSrcA = 1
NSrcB = 1
ALUSrc = 1
ALUOp = "somma"
RegDst = 0
MemRead
MemToReg = 1
RegWrite
RegWrite2