UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

La pipeline

Luigi Palopoli

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Ripartiamo da questo....

- Abbiamo visto come realizzare un semplice processore che esegue le istruzioni in un ciclo
- Questo non si fa piu' perche':
  - A dettare il clock sono le istruzioni piu' lente (accesso alla memoria)
  - Se si mettono istruzioni piu' complesse di quelle che abbiamo visto, le cose peggiorano ancora di piu' (esempio istruzioni floating point)
  - Non si riesce a fare ottimizzazioni aggressive sulle cose fatte piu' di frequente.

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Che fare allora?

- Il cosa fare ce lo insegno' Henry Ford con la catena di montaggio (assembly line):

"If everyone is moving forward together, then success takes care of itself."

"Coming together is a beginning. Keeping together is progress. Working together is success."

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Un esempio

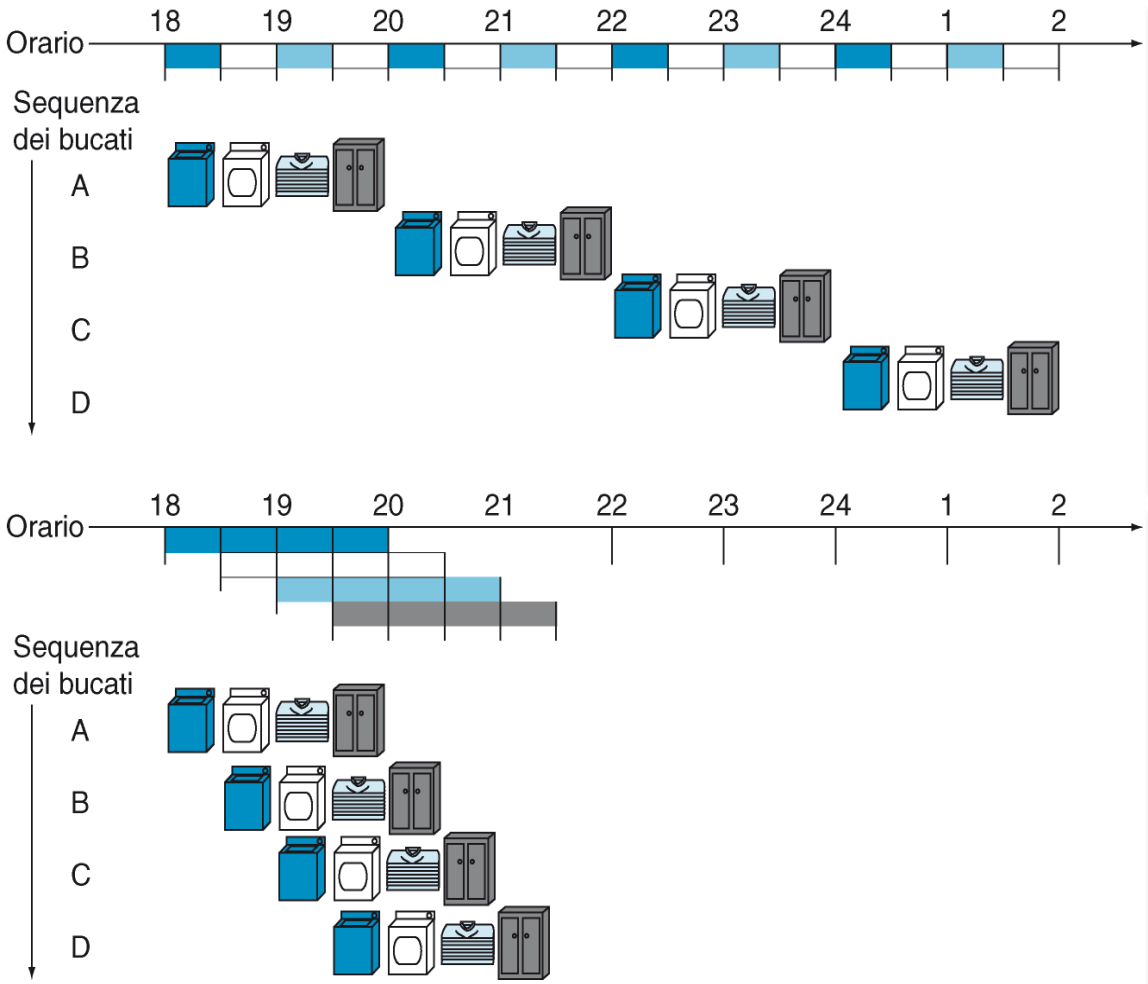
- Supponiamo di dover fare il bucato e che questo consista nelle seguenti attivita':

1. Mettere la biancheria nella lavatrice
2. Terminato il lavaggio mettere i panni nell'asciugatrice
3. Terminata l'asciugatura mettere I panni sull'asse da stiro e procedere alla stiratura
4. Finita la stiratura chiedere al proprio co-inquilino di riporre I panni

- Abbiamo un grande vantaggio se mentre la lavatrice lava un bucato, l'aciugatrice ne asciuga un altro e io ne stiro un altro ancora

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Una rappresentazione grafica



Supponendo che ciascuna fase duri 30 minuti, Il ciclo di lavaggio dura due ore (tra le 18 e el 20 per il primo).
Con il secondo metodo fino alle 21.30 ne faccio 4.

Notare che il tempo per fare un bucato (tempo di risposta) rimane invariato (due ore). Il throughput passa da 1/4 a 4/7 (miglioramento di 2.3). Se gli diamo tempo si arriva a un fattore 4

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Andiamo su un esempio piu' serio

- Tornando al MIPS, le fasi di esecuzione di un'istruzione sono le seguenti:

1. Prelievo dell'istruzione dalla memoria
2. Lettura dei registri e decodifica dell'istruzione (possibile per il formato regolare delle istruzioni MIPS)
3. Esecuzione di un'operazione (tipo R) o calcolo di un indirizzo
4. Accesso a un'operando nella memoria dati
5. Scrittura del risultato in un registro

- Conseguentemente useremo una pipeline a cinque stadi

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Un esempio

- Limitiamoci per il momento a una pipeline che sia in grado di effettuare le seguenti operazioni:
  - LOAD (lw), STORE (sw)
  - AND (and), OR (or)
  - Set less than (slt)
  - Branch on equal (beq)

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Tempi

- I tempi richiesti per le varie fasi sono I seguenti:

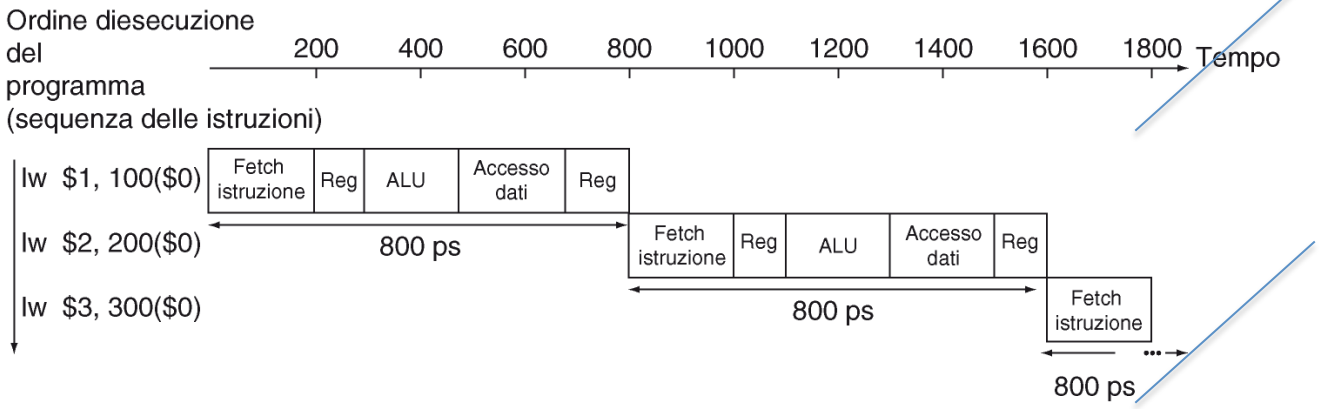
Classe dell'istruzione	Lettura dell'istruzione	Lettura dei registri	Operazione con la ALU	Accesso ai dati in memoria	Scrittura del register file	Tempo totale
Load word (`lw`)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (`sw`)	200 ps	100 ps	200 ps	200 ps		700 ps
Formato R (`add`, `sub`, AND, OR, `slt`)	200 ps	100 ps	200 ps		100 ps	600 ps
Salto condiziona-to (beq)	200 ps	100 ps	200 ps			500 ps

- L'istruzione piu' lenta e' la lw. Quindi ci si deve adeguare ad essa per la scelta del clock

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

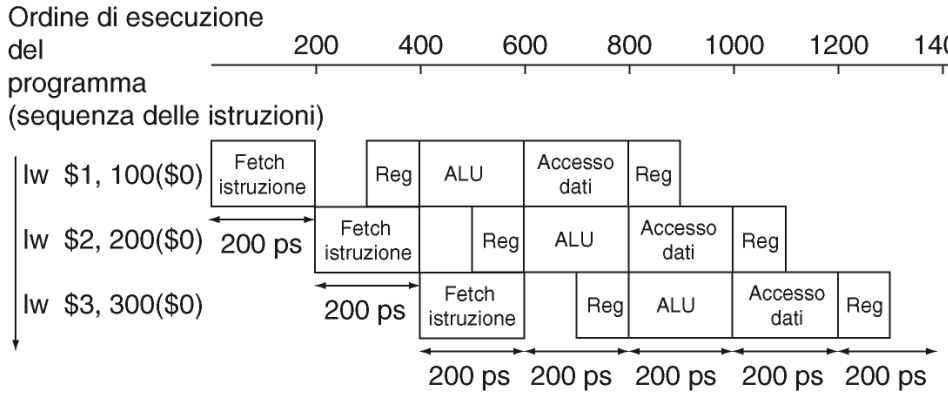
Varie implementazioni

- Diagramma temporale per seguenza istruzioni:



Implementazione a singolo ciclo

Implementazione a pipeline



Tempo tra fine prima a fine quarta quarta istruzioni passa da 2400 (3*800) a 1400 ps.

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Confronto di prestazioni

- Un confronto di prestazioni puo' essere fatta con la seguente formula (valida se la pipeline opera in condizioni ideali)

$$\text{Tempo tra due istruzioni con pipeline} = \frac{\text{Tempo tra due istruzioni senza pipeline}}{\text{Numero di stadi della pipeline}}$$

- La formula suggerisce che con cinque stadi dovremmo arrivare a un tempo tra due istruzioni (inverso del trhoughput) che dovrebbe essere 1/5.

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Osservazioni

- Nell'esempio precedente siamo passati da 2400 a 1400 (miglioramento 1.7) come mai non 5?
  - *Prima osservazione*: per ottenere una prestazione di 1/5 dovremmo portare il clock a 160 ps (non possibile) perche' ci sono alcune fasi che durano 200 ps (al piu' 4 volte)
    - *Seconda osservazione*: ovviamente il problema e' che abbiamo considerato poche istruzioni e non abbiamo fatto in tempo a riempire la pipeline.

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Comportamento al limite

- Se consideriamo molte piu' istruzioni (ad esempio **1000000**) per un totale di **1000003** si passa da un tempo di esecuzione di 1000003*800ps a un tempo di esecuzione di 1400 + 1000000*200ps.
- Se facciamo il rapport vediamo che l'incremento prestazionale in termini di *throughput* is avvicina al 400%

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Vantaggi del RISC

Vantaggi delle architetture RISC (a la MIPS per il pipelining)

- ***Primo vantaggio:*** tutte le istruzioni hanno la stessa lunghezza. Questo facilita di molto il prelievo (sempre una word)

- ***Secondo vantaggio***: I codici degli operandi sono in posizione fissa. Questo permette di accedervi leggendo il register file in parallelo con la decodifica dell'istruzione

- ***Terzo vantaggio***: gli operandi residenti in memoria sono possibili solo per lw e sw. Cio' permette di usare la ALU per il calcolo di indirizzi (cosa che non sarebbe possibile se dovessimo usare le ALU in due fasi della stessa istruzione)

- ***Quarto vantaggio***: l'uso di accessi allineati fa si che gli accessi in memoria avvenfono sempre in un ciclo di trasferimento (impegnando un solo stadio della pipeline)

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Hazard

- In condizioni normali la pipeline permette di eseguire una istruzione per ciclo di clock.
- Alle volte questo non e' possibile per il verificarsi di condizioni critiche (detti hazard).
- Passiamo in rassegna alcuni tipologie di hazard.

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Hazard Strutturali

- Una condizione di hazard strutturale e' una per la quale l'architettura dell'elaboratore rende impossibile l'esecuzione di alcune sequenze di istruzioni in pipeline.
- Ad esempio, se io disponessi di un'unica memoria, non potrei nello stesso ciclo, caricare istruzioni e memorizzare (o prelevare) operandi dalla memoria.

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Hazard sui dati

- Questo tipo di hazard si verifica quando la pipeline deve essere messa in stallo per ottenere delle informazioni dagli stadi precedenti
- Nell'esempio della striatura, se mi accogo che manca un calzino, devo interrompere la stiratura dell'altro e andarlo a cercare (bloccando anche le fasi precedenti).
- Nel caso del MIPS, consideriamo la seguente sequenza

```
add $s0, $t0, $t1
sub $t2, $s0, $t3
```

- Il problema e' che s0 viene memorizzato nella quinta fase, mentre la sub dopo ne ha bisogno nella seconda fase....quindi e' costretta ad aspettare

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

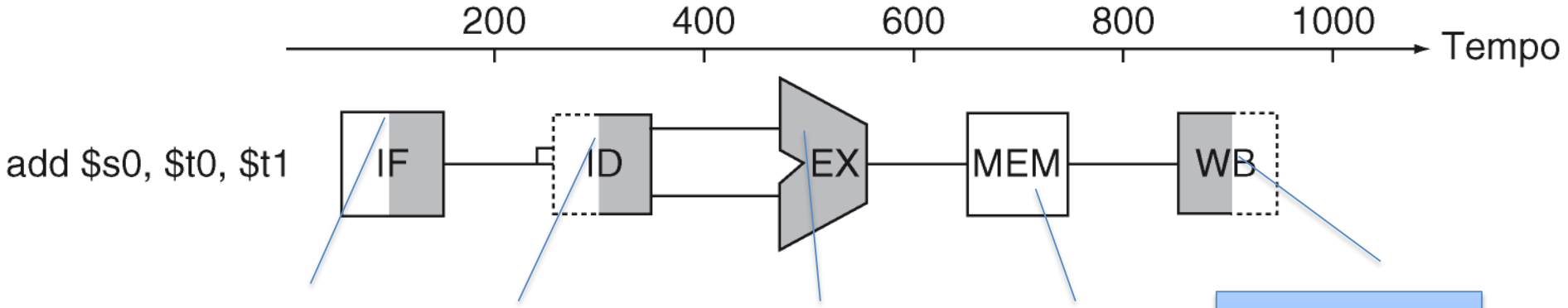
Come risolverlo?

- L'hazard precedente blocca il completamento della second istruzione per tre cicli di clock
- In certi casi possiamo cavarcela a livello di compilazione invertendo alcune istruzioni
- Tuttavia il compilatore puo' risolvere il problema solo in alcuni casi
- In generale, e' utile osservare che non occorre aspettare di memorizzare il risultato

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Torniamo all'esempio

- Una rappresentazione grafica per l'operazione della pipeline su una delle istruzioni e' la seguente:



Instruction fetch (qudratino memoria istr.)

Instruction decode (qudaratino spezzato register file)

Instruction Execute

Accesso alla memoria

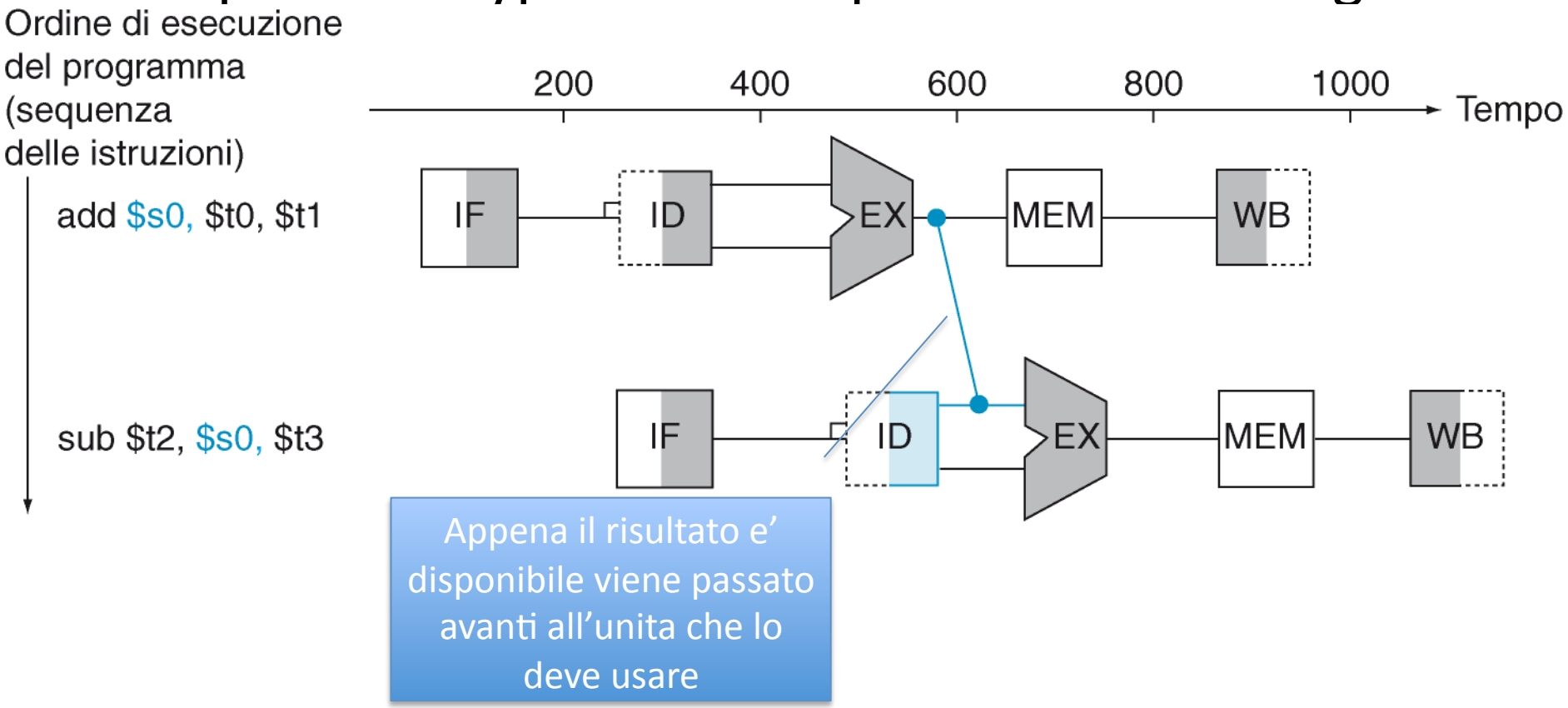
Write back (scrittura risulato nel register file)

Ombra = unita' usata per istruzione

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Operand forwarding

- L'operand forwarding (o propagazione) o anche bypass viene usato per rendere il risultato disponibile bypassando l'operazione di storage



UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Hazard sul controllo

- Il terzo tipo di hazard riguarda il controllo (sostanzialmente i salti condizionati)
- Torniamo all'esempio del bucato
  - Supponiamo che, a seconda del livello di sporco si voglia decidere per un lavaggio aggressivo
  - Quello che dovrei fare e' verificare la condizione dei panni all'uscita dell'asciugatrice e su questa base cambiare le impostazioni
  - ma nel far questo si blocca la pipeline (fino a che non ho finito l'asciugatura non posso procedere al lavaggio della prossima mandata)

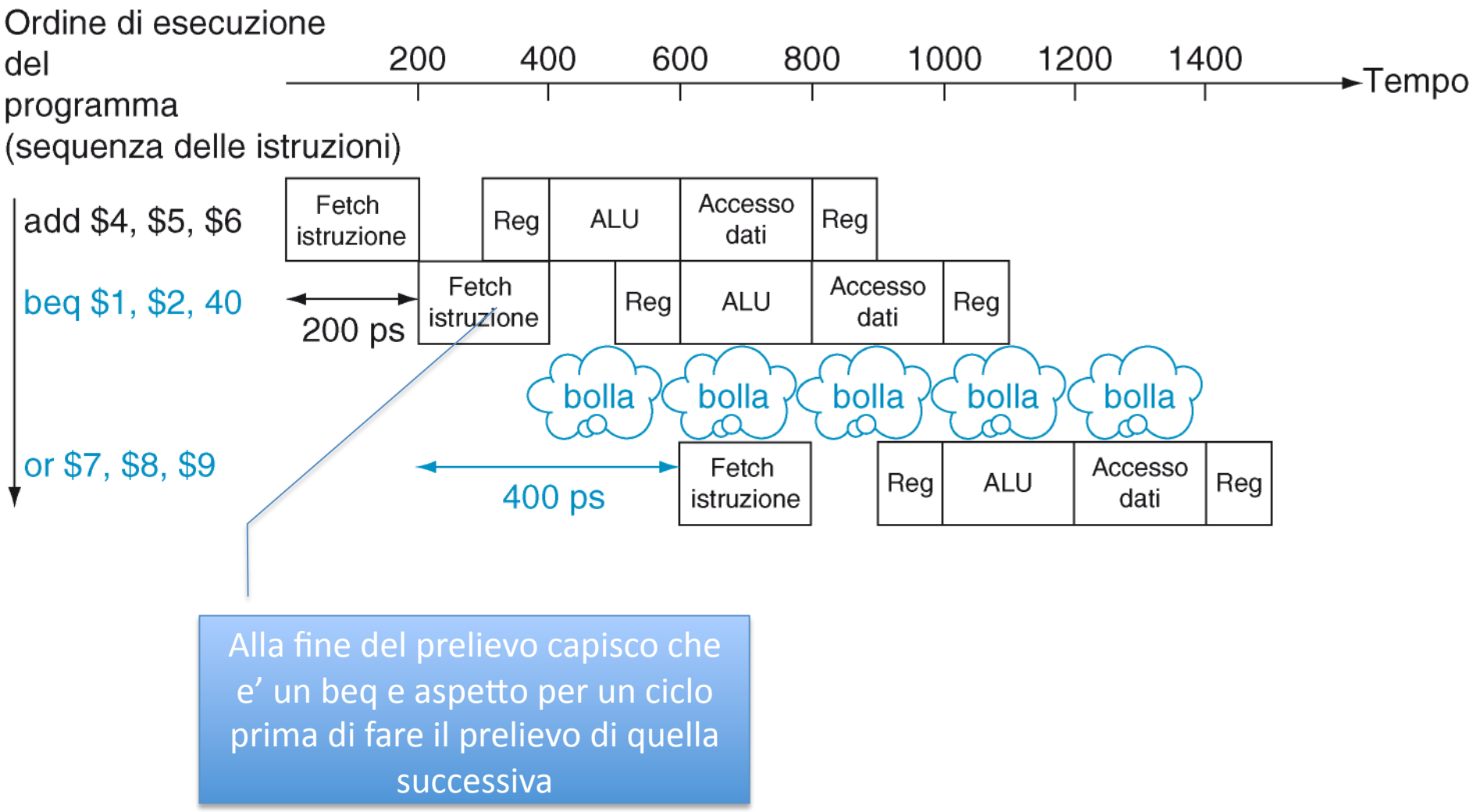
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Salto condizionato

- Il caso simile a quello del bucato si presenta con I salti condizionati
- Supponiamo di avere un circuito moto sofisticato che ci permettta di calcolare l'indirizzo di salto gia' al secondo stadio
- Comunque, dobbiamo bloccare la pipeline per uno o due cicli

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Esempio



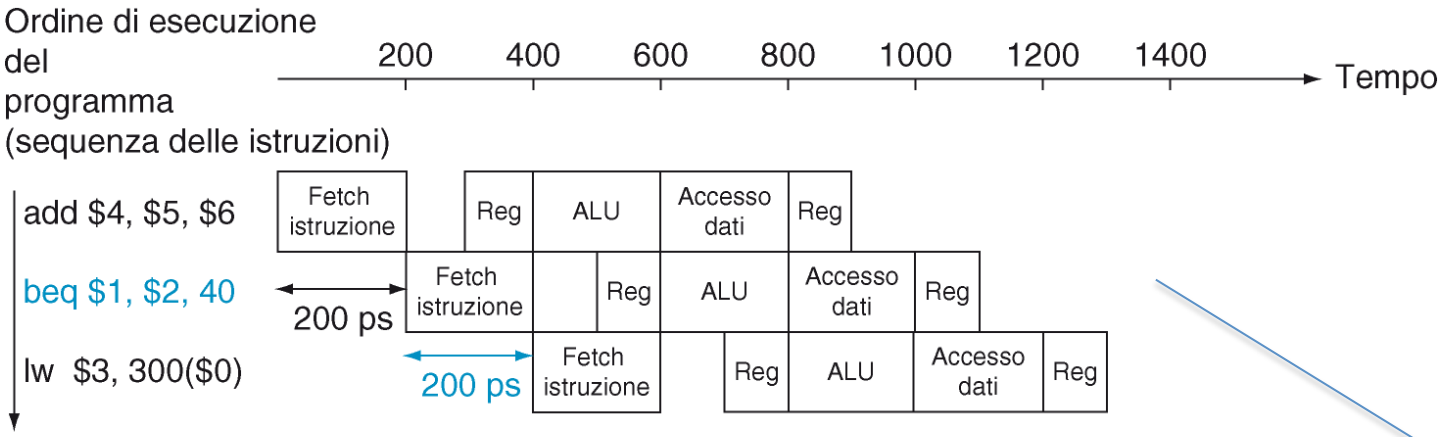
UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Predizione del salto

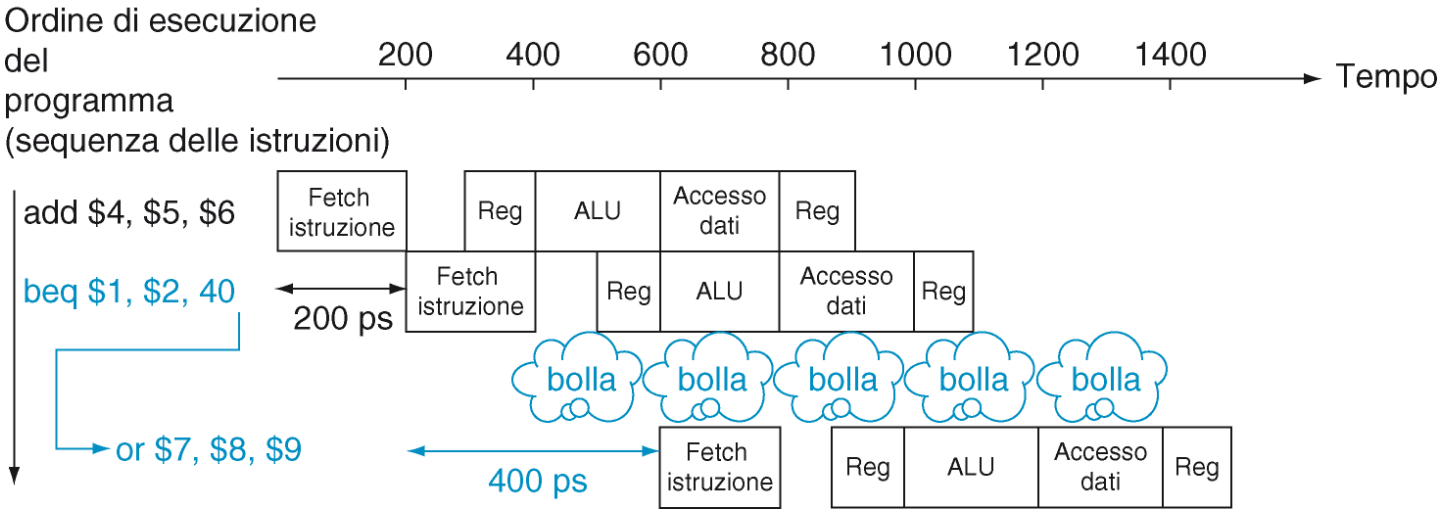
- Se la pipeline e' piu' lunga generare questo stallo e' un costo troppo lungo
- Quello che si fa e' avere dei circuiti che prevedano i salti
- Ad esempio, nel caso precedente si puo' assumere che il salto non venga preso e poi corregersi in caso contrario

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Esempio precedente



Si assume che non venga preso



UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Circuiteria di branch prediction

- L'esempio che abbiamo visto primo non e' particolarmente sofisticato (finisce con il funzionare bene solo se il branch non viene preso)
- Esistono circuiterie piu' sofisticate che permettono di memorizzare l'esisto del branch precedente e assumere che il comportamento si mantenga coerente.