# Esercizio: Dipendenze

***Dipendenza dai dati*** : per la quale l'istruzione *j* dipende dall'istruzione *i* se *i* produce, direttamente o transitivamente (ossia tramite una o più istruzioni intermedie) un risultato richiesto da *j*.

***Dipendenza dal controllo*** : la quale determina l'ordinamento di una istruzione rispetto ad un salto condizionale, così che essa esegua solo quando dovuto rispetto all'esecuzione del salto.

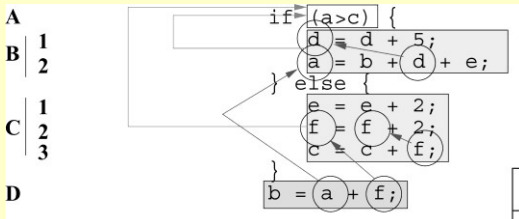
***Dipendenza dai nomi*** : la quale ha luogo allorchè due istruzioni, tra le quali non vi sia flusso di dati, usano lo stesso registro o la stessa locazione di memoria.

Si illustrino tutte le dipendenze presenti nel seguente frammento di programma in linguaggio C, assumendo che il programma non faccia riferimento ad altri dati, che tutti i valori siano definiti prima dell'uso, e che soltanto `b` e `c` siano usati successivamente alle istruzioni date:

```
if (a>c) {
    d = d + 5;
    a = b + d + e; }
else {
    e = e + 2;
    f = f + 2;
    c = c + f; }
b = a + f;
```

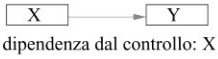


# Soluzione




| | dipendenze | |
|---|---|---|
| **dal controllo** | **dai dati** | **dai nomi** |
| B → A | B.2 → B.1 | |
| C → A | C.3 → C.2 | |
| | D → C.2,B.1 (via B.2) | |

legenda:




dipendenza dal controllo: X dipende da Y


dipendenza dai dati: X dipende da Y

# Esercizio: valutazione delle prestazioni

- Si considerino le seguenti statistiche:
  - 15% delle istruzioni sono di salto condizionale
  - 1% delle istruzioni sono di salto incondizionale
  - Il 60% delle istruzioni di salto condizionale hanno la condizione soddisfatta (prese)
- …ed una pipeline a 4 stadi (IF, ID, EI, WO) per cui:
  - i salti incondizionati sono risolti (identificazione salto e calcolo indirizzo target) alla fine del secondo stadio (ID)
  - i salti condizionati sono risolti (identificazione salto, calcolo indirizzo target e calcolo condizione) alla fine del terzo stadio (EI)
  - il primo stadio (IF) è indipendente dagli altri
  - ogni stadio impiega 1 ciclo di clock
- inoltre si assuma che non ci siano altre istruzioni che possano mandare in stallo la pipeline e che si predica di non saltare in caso di salto condizionale

**Domanda:**
**calcolare quanto più veloce, a regime, sarebbe la pipeline senza gli stalli introdotti dai salti**

Aiuto: fattore di velocizzazione di una pipeline a k stadi, a regime, in funzione del numero di stalli:

$$S_k = \frac{1}{1 + frazione\_cicli\_stallo}\, k$$

# Soluzione: valutazione delle prestazioni

- Per rispondere alla domanda bisogna calcolare il rapporto tra le prestazioni di una pipeline a 4 stadi senza stalli con le prestazioni della pipeline con ritardi
- Le prestazioni di una pipeline a 4 stadi senza ritardi si ottengono considerando la formula data con $k$=4 e 0 cicli di stallo:

$$\frac{1}{1+0} 4 = 4$$

- Per calcolare le prestazioni in presenza di stalli bisogna calcolare:
  - la probabilità di eseguire una delle istruzioni di salto

    salto incondizionato → **0,01** perché 1 su 100 è un salto incondizionato
    salto condizionato preso → 0,15*0,6 = **0,09** perché 15 istr. su 100, e il 60% salta
    salto condizionato non preso → 0,15*0,4 = **0,06** perché 15 istr. su 100, e il 40% non salta

  - la frazione di cicli di stallo per tipo di istruzione di salto

    vedi prossimi lucidi

# Soluzione: valutazione delle prestazioni

- Stalli per salto incondizionato (salta all'istruzione con indirizzo *j*)

cicli clock

| istr. eseguita | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| jump | IF | ID | EI | WO | | |
| *i + 1* | | ~~IF~~ | *(qui la pipeline è "svuotata")* | | | |
| istr. target | | | IF | ID | EI | … |
| *j + 1* | | | | IF | ID | … |
| *j + 2* | | | | | IF | … |

quindi si ha **1 ciclo** di "stallo" (non è un vero e proprio stallo: la pipeline è svuotata, quindi si esegue un IF inutile)



# Soluzione: valutazione delle prestazioni

- Stalli per salto condizionato **preso** (salta all'istruzione con indirizzo *j*)

cicli clock

| istr. eseguita | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| branch | IF | ID | EI | WO | | |
| *i + 1* | | IF | ~~ID~~ | *(qui la pipeline è "svuotata")* | | |
| *i + 2* | | | ~~IF~~ | *(qui la pipeline è "svuotata")* | | |
| istr. target | | | | IF | … | |
| *j + 1* | | | | | IF | … |

quindi si hanno **2 cicli** di "stallo"

- Stalli per salto condizionato **non preso**

cicli clock

| istr. eseguita | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| branch | IF | ID | EI | WO | | |
| *i + 1* | | IF | ID | EI | … | |
| *i + 2* | | | IF | ID | … | |
| *i + 3* | | | | IF | … | |

quindi si hanno **0 cicli** di "stallo"

# Soluzione: valutazione delle prestazioni

Rappresentazione alternativa

- Stalli per salto condizionato **preso** (salta all'istruzione con indirizzo *j*)

| stadi \ cicli clock | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| IF | branch | *i+1* | *i+2* | *istr. target* | *j+1* | *j+2* |
| ID | | branch | *i+1* | *bubble* | *istr. target* | *j+1* |
| EI | | | branch | *bubble* | *bubble* | *istr. target* |
| WO | | | | … | | |

Si noti che ogni stadio "perde" 2 cicli di clock:

- IF carica le istruzioni con indirizzi *i+1* e *i+2* che poi non terminano l'esecuzione;
- ID decodifica l'istruzione con indirizzo *i+1* che non termina l'esecuzione e poi rimane inattiva durante il ciclo di clock 4 (*bubble*);
- EI (e successivamente WO) rimane inattiva durante i cicli di clock 4 e 5.



# Soluzione: valutazione delle prestazioni

- la frazione di cicli in cui si ha stallo è:

| | |
|---|---|
| prob_jump*stalli_jump | [**0,01*1**] |
| + | + |
| prob_branch_preso* stalli_branch_preso | [**0,09*2**] |
| + | + |
| prob_branch_non_preso* stalli_branch_non_preso | [**0,06*0**] |
| | = |
| | **0,19** |

- e quindi le prestazioni della pipeline con stalli è:

$$S_k = \frac{1}{1+0,19} 4 = 3,36$$