AXO – Architettura dei Calcolatori e Sistemi Operativi

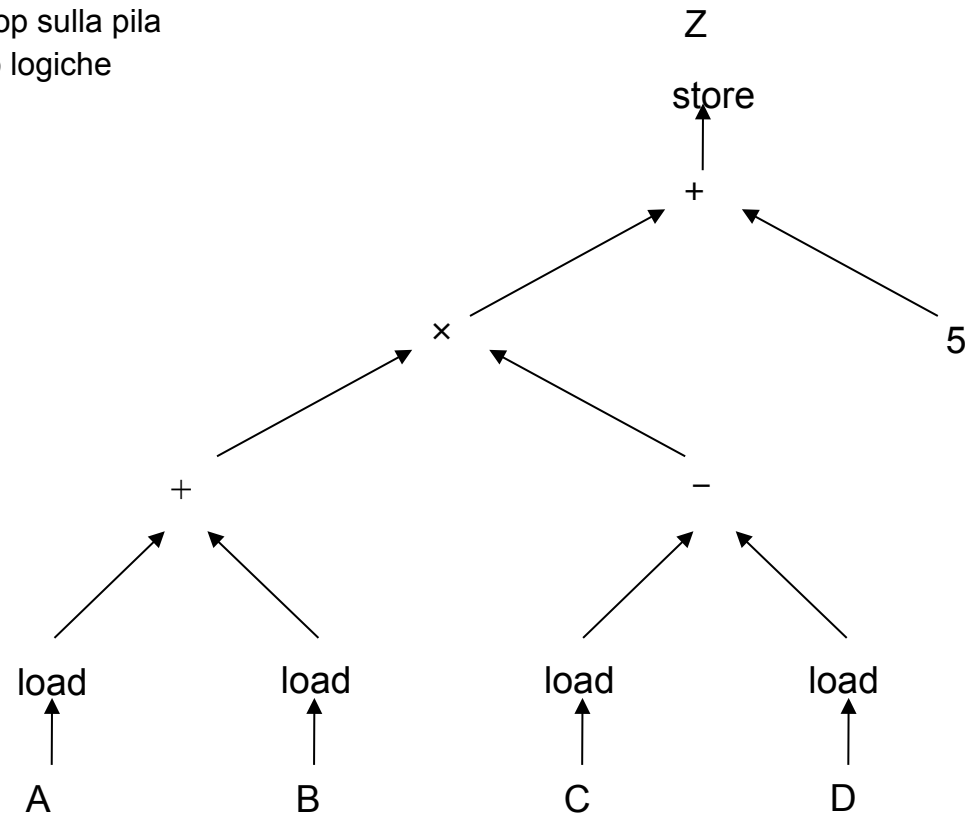
calcolo delle espressioni

algoritmo per il calcolo delle espressioni

- data un'espressione, per esempio
  - $Z = (A + B) \times (C - D) + 5$ o anche
  - $Z = A + f(B, C, D) \times E + g(-F)$
- ricavare il corrispondente programma assembler 68000 in grado di calcolarla
- **convenzioni utilizzate** (è una scelta)
  - il calcolo delle espressioni viene effettuato nei registri
  - il passaggio dei parametri avviene in pila e il valore restituito è memorizzato dal sottoprogramma sovrascrivendo il primo dei parametri passati
  - il chiamante deve deallocare l'area in pila dei parametri
- passi da eseguire
  1. ricavare **l'albero sintattico** dell'espressione
  2. **numerare i nodi interni** dell'albero secondo una visita *postordine*
  3. **coprire l'albero** con le tegole prefissate, che corrispondono a ben precise istruzioni macchina 68000
  4. **allocare i registri** (nell'ordine di numerazione, senza ottimizzazioni, rispettando i vincoli)
  5. **codificare il programma** in assembler seguendo l'ordine di numerazione dei nodi, con i registri allocati)

1 - determinare l'albero sintattico

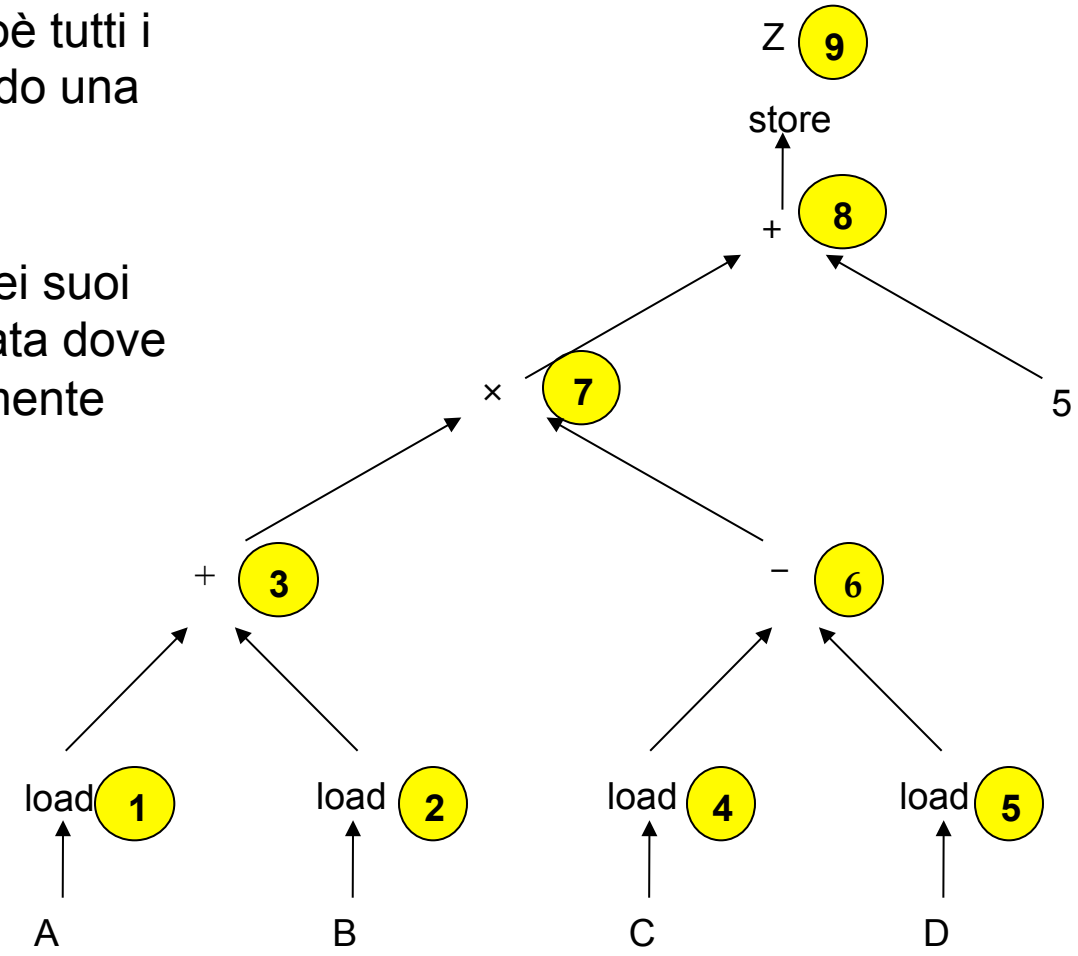
- espressione di esempio: $Z = (A + B) \times (C - D) + 5$
- determinazione dell'albero sintattico utilizzando gli elementi strutturali costituiti
  - da trasferimenti da/a memoria e trasferimenti tra registri
  - operazioni di push/pop sulla pila
  - operazioni aritmetico logiche
  - chiamate di funzioni



2 - numerare i nodi dell'albero

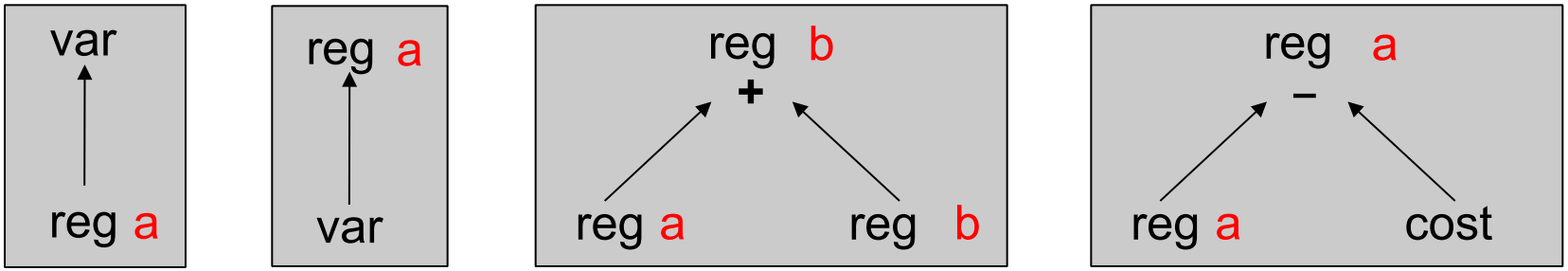
- numerare i **nodi interni** (cioè tutti i nodi tranne le foglie) secondo una visita in postordine

- dato un albero, una visita dei suoi nodi è una sequenza ordinata dove ogni nodo compare esattamente una volta



3 - coprire l'albero con tegole

- avete a disposizione un insieme di tegole prefissato (vedere prossime pagine), fatte per esempio come segue



- ogni tegola corrisponde ad una ben precisa istruzione (o sequenza di istruzioni)
- disponete le tegole in modo che combacino con i nodi dell'albero
- ogni nodo interno (salvo la radice) sia coperto da due tegole
- le tegole sono parzialmente sovrapposte; le zone sovrapposte **devono** combaciare
- i nomi dei registri sono utilizzati nella fase successiva di allocazione dei registri

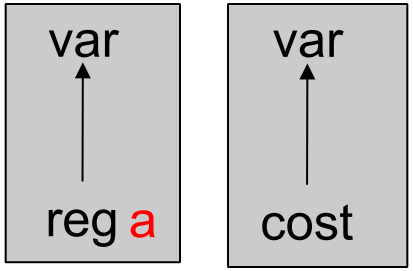
tegole da usare per la copertura - a

per le istruzioni di trasferimento dati

`MOVE Da, var`

`MOVE #cost, var`

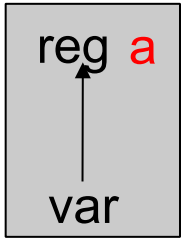
equivale a store

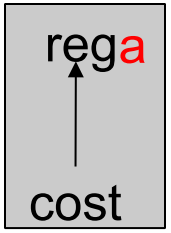


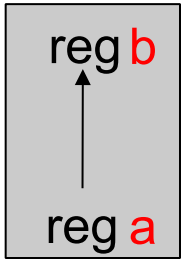
`MOVE var, Da`

`MOVE #cost, Da`

equivale a load





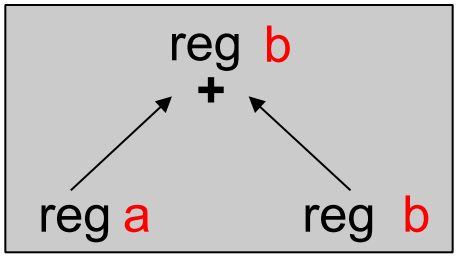


`MOVE Da, Db`

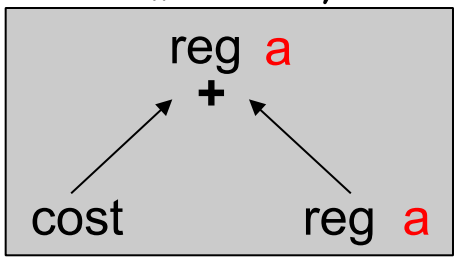
tegole da usare per la copertura - b

operazioni aritmetiche

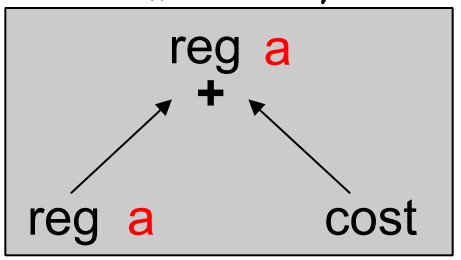
ADD Da, Db



ADD #cost, Da

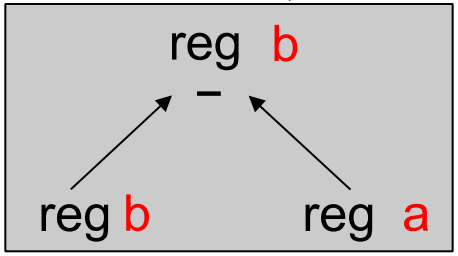


ADD #cost, Da



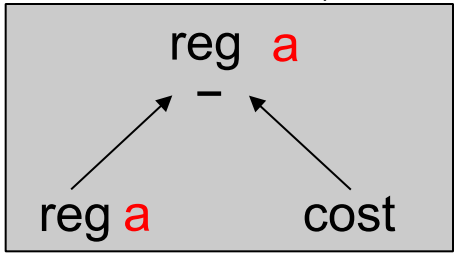
queste due tegole hanno la stessa traduzione assembly
in virtù della commutatività dell'operatore “+”

SUB Da, Db



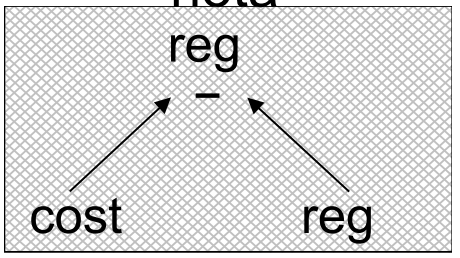
attenzione all'ordine
degli operandi !

SUB #cost, Da



attenzione all'ordine
degli operandi !

nota



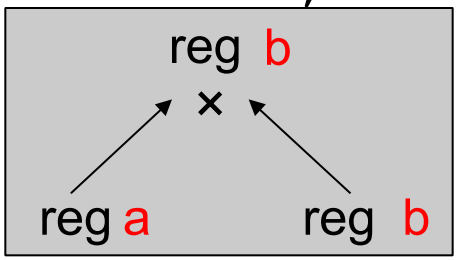
questa tegola non esiste !

se si presenta questo sottoalbero,
spostare “cost” in un registro e poi
usare la prima tegola a sinistra

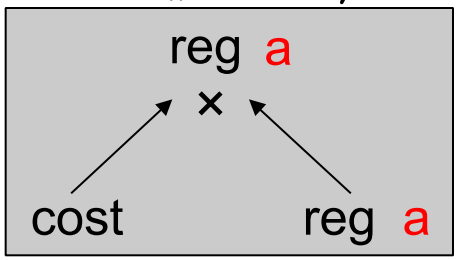
tegole da usare per la copertura - c

ancora operazioni aritmetiche (come prima)

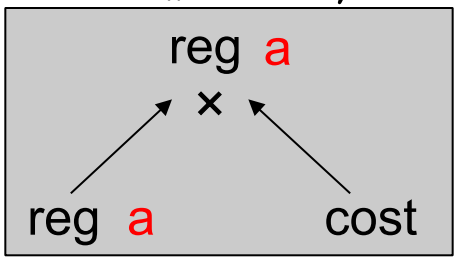
`MULS Da, Db`



`MULS #cost, Da`

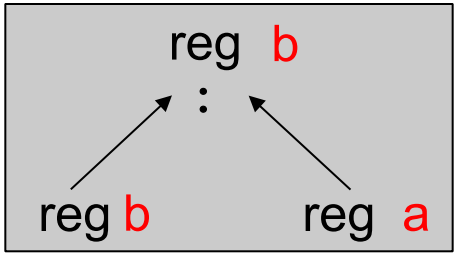


`MULS #cost, Da`



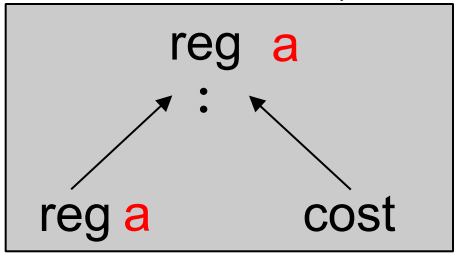
queste due tegole hanno la stessa traduzione assembly in virtù della commutatività dell'operatore "´"

`DIVS Da, Db`



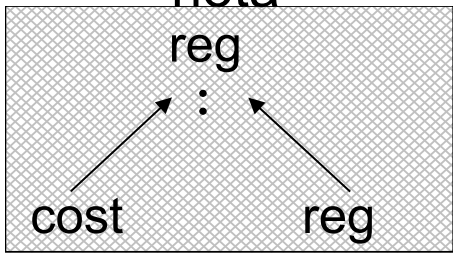
attenzione all'ordine degli operandi !

`DIVS #cost, Da`



attenzione all'ordine degli operandi !

nota



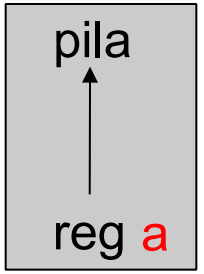
questa tegola non esiste !

se si presenta questo sottoalbero, spostare cost in un registro e poi usare la prima tegola a sinistra

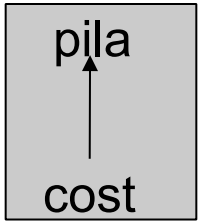
tegole da usare per la copertura - d

push dei parametri di funzione

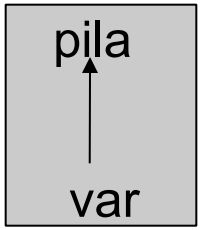
```
MOVE Da, -(A7)
```




```
MOVE #cost, -(A7)
```



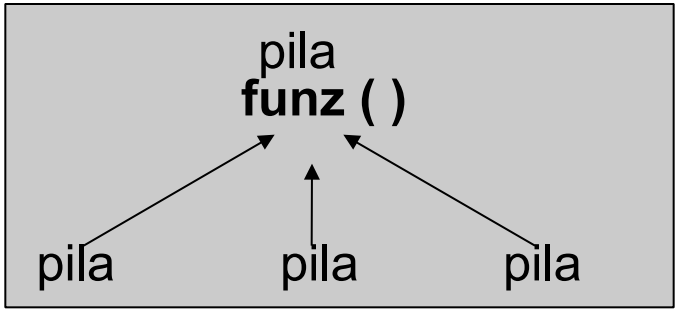

```
MOVE var, -(A7)
```



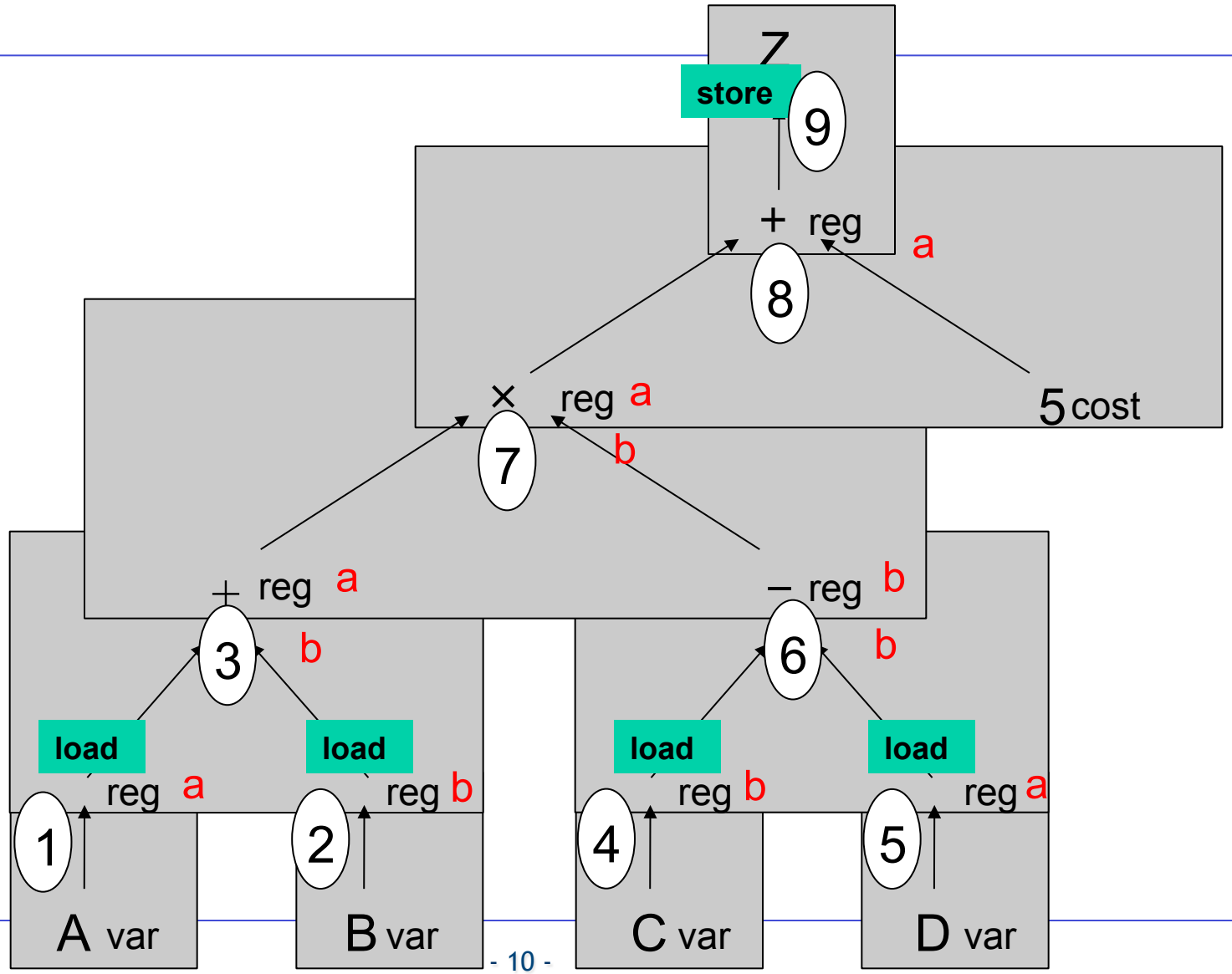
chiamata di funzione

```
BSR    funz
ADDA   spi., A7
```

la ADDA dealloca l'area parametri e posiziona SP al valore restituito, che poi va spilato

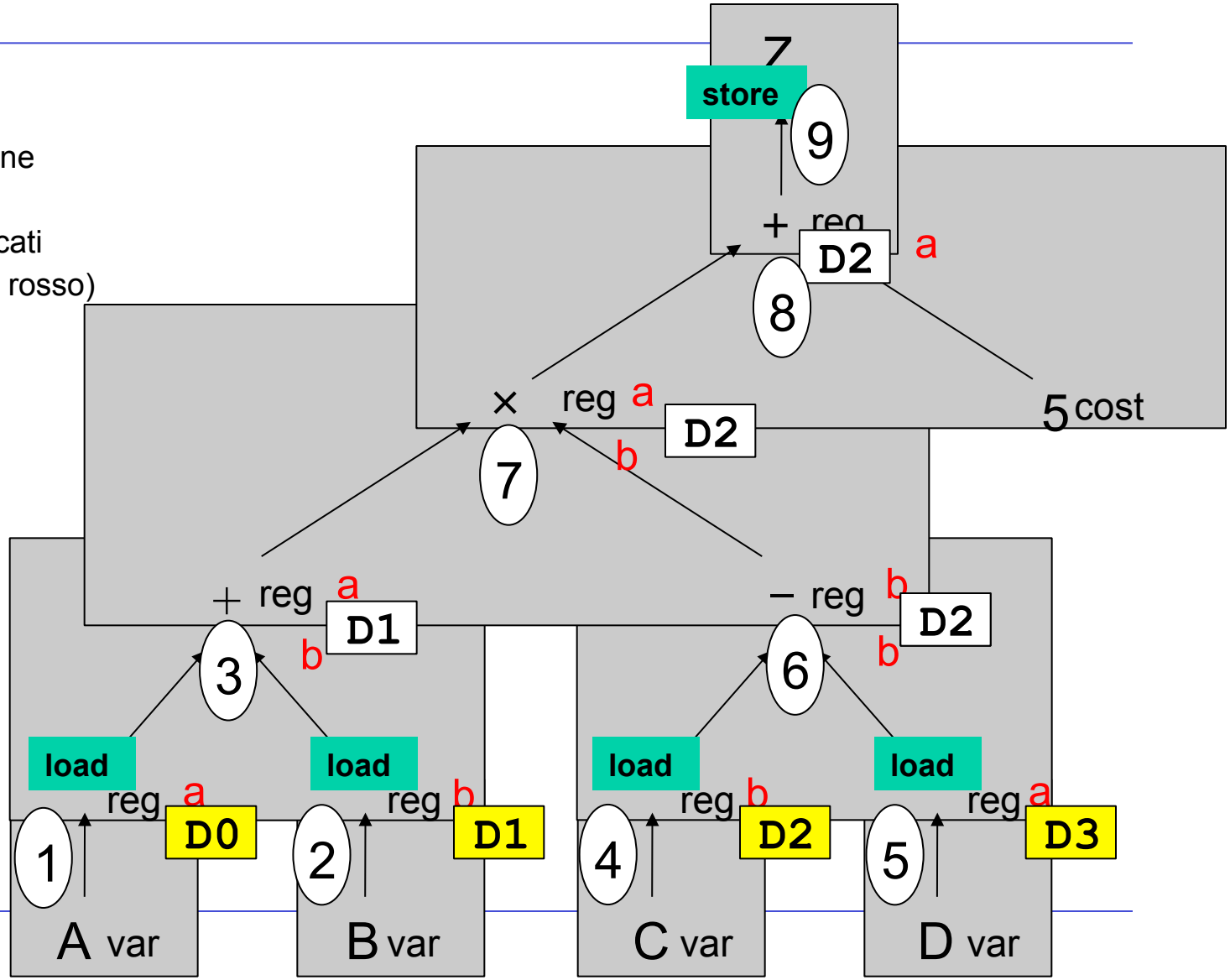


3 - coprire l'albero con tegole



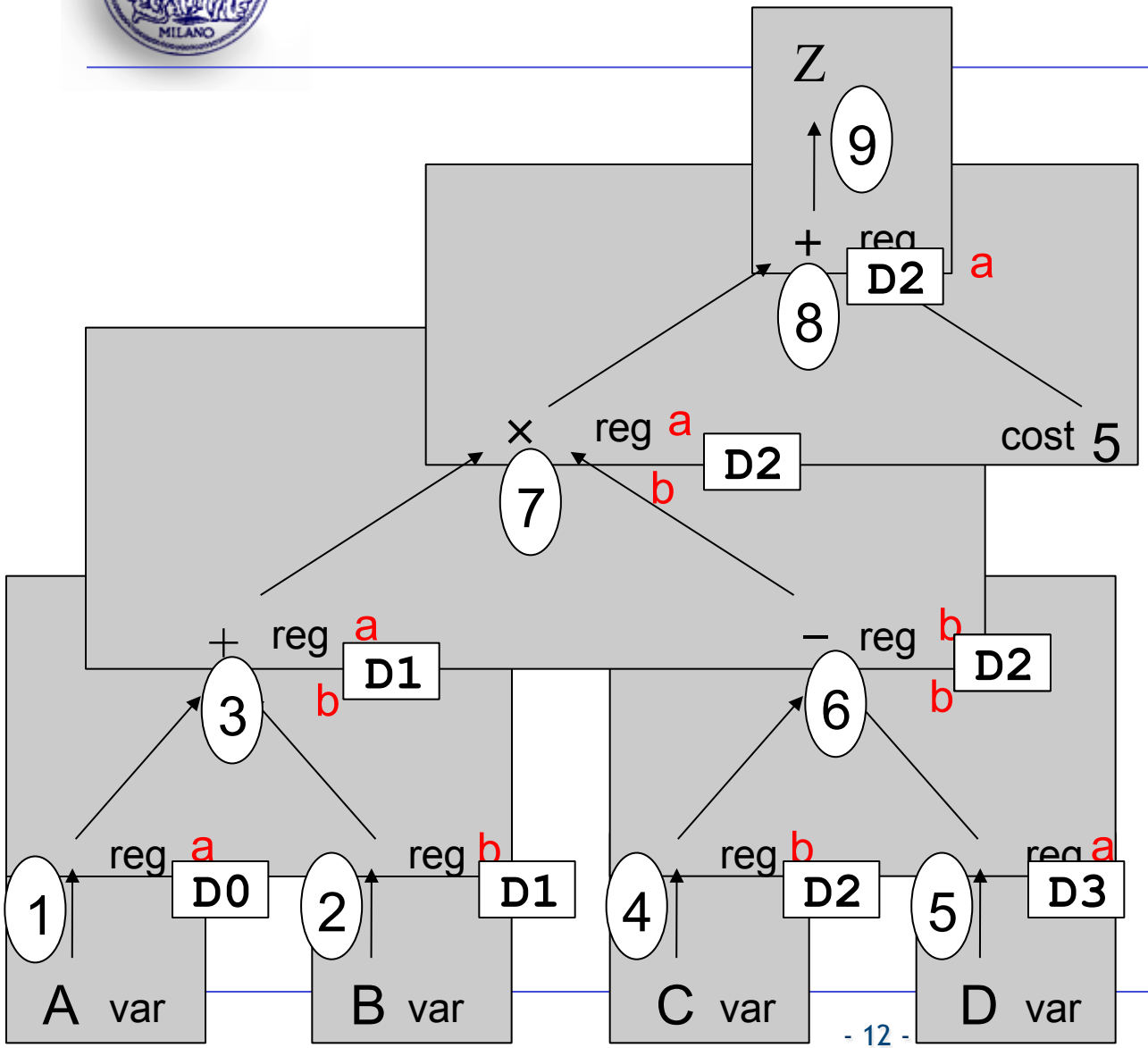
4 - allocare (o schedulare) i registri

allocare i registri
- nell'ordine di numerazione del passo precedente
- rispettando i vincoli indicati nelle tegole (segnalati in rosso)



nota: i soli registri in giallo sono allocati liberamente, gli altri sono vincolati

5 - codificare il programma



codificare in assembler

```
1. MOVE.L A,  D0
2. MOVE.L B,  D1
3. ADD.L  D0, D1
4. MOVE.L C,  D2
5. MOVE.L D,  D3
6. SUB.L  D3, D2
7. MULS   D1, D2
8. ADD.L  #5, D2
9. MOVE.L D2, Z
```

programma finale

realizzare il programma completo che calcola

$$Z = (A + B) \times (C - D) + 5$$

con i valori seguenti

A	= 1
B	= 2
C	= 3
D	= -4
Z	(calcolata) = 26

```
// allocazione delle variabili, parola doppia
      ORG    $1000
A:    DS.L   1
B:    DS.L   1
C:    DS.L   1
D:    DS.L   1
Z:    DS.L   1
START:     ORG    $1000
      // inizializzazione delle variabili
      MOVE.L #1,  A
      MOVE.L #2,  B
      MOVE.L #3,  C
      MOVE.L #-4, D
      MOVE.L #0,  Z
      // calcolo dell'espressione
      MOVE.L A,  D0
      MOVE.L B,  D1
      ADD.L  D0, D1
      MOVE.L C,  D2
      MOVE.L D,  D3
      SUB.L  D3, D2
      MULS   D1, D2 // attn: 16 + 16 bit
      ADD.L  #5, D2
      MOVE.L D2, Z
      STOP   #$2000
      END    START
```