

architettura degli elaboratori lezione 7


Appunti di Davide Scarlata 2024/2025

 **Prof:** Claudio Schifanella

 **Mail:** claudio.schifanella@unito.it

 **Corso:** C

 **[Moodle Unito](#)**

 **Data:** 18/03/2025

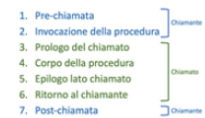
convenzioni di chiamata

Convenzione nell'uso e salvataggio dei registri

- Necessità di stabilire delle convenzioni
- I registri `x10-x17` (`a0-a7`), `x5-x7` e `x28-x31` (`t0-t6`)
 - possono essere modificati dal chiamato senza nessun meccanismo di ripristino
 - Il chiamante se necessario dovrà salvare i valori dei registri prima dell'invocazione della procedura
- I registri `x1` (`ra`), `x2` (`sp`), `x3` (`gp`), `x4` (`tp`), `x8` (`fp/s0`), `x9` e `x18-x27` (`s1-s11`)
 - Se modificati dal chiamato devono essere salvati e poi ripristinati prima del ritorno al chiamante
 - Il chiamante non è tenuto al loro salvataggio e ripristino

Vantaggio: si evitano scritture e letture inutili

Le fasi di una invocazione di procedura



Fase 1 – Pre-chiamata del chiamante

- Eventuale salvataggio registri da preservare nel chiamante
 - Si assume che `x10-x17` (`a0-a7`), `x5-x7` e `x28-x31` (`t0-t6`), possano essere sovrascritti dal chiamato
 - se li si vuole preservare vanno salvati nello stack (dal chiamante) – vedi caso 1
 - il caso 2 mostra un caso in cui non è necessario salvare il contenuto del registro associato alla variabile `f`
- Preparazione degli argomenti della funzione
 - I primi 8 argomenti vengono posti in `x10-x17`, ovvero `a0-a7` (nuovi valori)
 - Gli eventuali altri argomenti oltre l'ottavo vanno salvati nello stack (`EXTRA_ARGS`), così che si trovino subito sopra il frame della funzione chiamata

```
int somma(int x, int y) {  
    x=x+y;  
    return x;  
}  
...  
f=f+1;  
risultato=somma(f,g);  
printf("%d", f);  
return;
```

1

```
int somma(int x, int y) {  
    x=x+y;  
    return x;  
}  
...  
f=f+1;  
risultato=somma(f,g);  
return;
```

2

Le fasi di una invocazione di procedura



Fase 2 – Invocazione della procedura

- Istruzione `jal NOME_PROCEDURA`

Fase 3 – Prologo lato chiamato

- Eventuale allocazione del call-frame sullo stack (aggiornare `sp`)
 - Salvataggio di `x1 (ra)` nel caso in cui la procedura non sia foglia
 - Salvataggio di `x8 (fp)`, solo se utilizzato all'interno della procedura
 - Salvataggio di `x9` e `x18-x27 (s1-s11)` se utilizzati all'interno della procedura (il chiamante si aspetta di trovarli intatti)
 - Salvataggio degli argomenti `x10-x17 (a0-a7)` solo se la funzione li riuserà successivamente a ulteriori chiamate a funzione (procedure annidate) (nota: `x10-x17` sono trattati come i registri temporanei – possono essere sovrascritti)
- Eventuale inizializzazione di `fp`: punta al nuovo call-frame

Le fasi di una invocazione di procedura

Fase 4 – Corpo della procedura

- Istruzioni che implementano il corpo della procedura

Fase 5 – Epilogo lato chiamato

- Se deve essere restituito un valore dalla funzione
 - Tale valore viene posto in `x10` (e `x11`) ovvero `a0-a1`
- I registri (se salvati) devono essere ripristinati
 - `x9` e `x18-x27 (s1-s11)`
 - `x1 (ra)`
 - `x8 (fp)`
- Notare che `sp` deve solo essere aumentato di opportuno offset (lo stesso sottratto nella Fase 3)

Le fasi di una invocazione di procedura

2. Invocazione della procedura
3. Prologo del chiamato
4. Corpo della procedura
5. Epilogo lato chiamato
6. Ritorno al chiamante
7. Post-chiamata

Fase 6 – Ritorno al chiamante

- Istruzione `jalr x0, 0(x1)` (o pseudo-istruzione `jr ra`)

Fase 7 – Post-chiamata lato chiamante

- Eventuale uso del risultato della funzione (in `x10` e `x11`, cioè `a0-a1`)
- Ripristino dei valori `x5-x7` e `x28-x31` (`t0-t6`), `x10-x17` (`a0-a7`) vecchi, eventualmente salvati

I registri e convenzioni sul loro uso

Registro	Nome	Utilizzo
<code>x0</code>	zero	Costante zero
<code>x1</code>	ra	Return address
<code>x2</code>	sp	Stack pointer
<code>x3</code>	gp	Global pointer
<code>x4</code>	tp	Puntatore a thread
<code>x8</code>	s0 / fp	Frame pointer (il contenuto va preservato se utilizzato dalla procedura chiamata)
<code>x10-x11</code>	a0-a1	Passaggio di parametri nelle procedure e valori di ritorno
<code>x12-x17</code>	a2-a7	Passaggio di parametri nelle procedure
<code>x5-x7</code> <code>x28-x31</code>	<code>t0-t2</code> <code>t3-t6</code>	Registri temporanei, non salvati in caso di chiamata
<code>x9</code> <code>x18-x27</code>	<code>s1</code> <code>s2-s11</code>	Registri da salvare: il contenuto va preservato se utilizzati dalla procedura chiamata

Le fasi di una invocazione di procedura schematicamente

Possiamo dividere l'invocazione di una procedura in 7 fasi:

1. Pre-chiamata
 2. Invocazione della procedura
 3. Prologo del chiamato
 4. Corpo della procedura
 5. Epilogo lato chiamato
 6. Ritorno al chiamante
 7. Post-chiamata
-
- The diagram uses brackets to group the phases into three categories:
- Chiamante** (blue bracket): 1. Pre-chiamata, 2. Invocazione della procedura, 7. Post-chiamata
 - Chiamato** (green bracket): 3. Prologo del chiamato, 4. Corpo della procedura, 5. Epilogo lato chiamato, 6. Ritorno al chiamante

Le fasi di una invocazione di procedura foglia

- Se una procedura è foglia, nel codice della procedura :
- non è necessario salvare e ripristinare il registro ra
- non è necessario salvare sullo stack i registri a0-a7 e t0-t6 (non c'è nessuna invocazione di procedura nella procedura chiamata)

```
sommaArray:
    li t0, 0 # t0 = somma
    li t1, 0 # t1 = contatore indice i
loop:
    bge t1, a1, fine      # i >= lunghezza, esci dal loop
    slli t2, t1, 2        # offset
    add t3, a0, t2        # indirizzo elemento i
    lw t4, 0(t3)          # Carica v[i]
    add t0, t0, t4        # Aggiorna la somma
    addi t1, t1, 1        # i++
    j loop               # Ripeti il ciclo
fine:
    mv a0, t0            # Restituisci il risultato in a0
    jr ra               # Ritorna

# funzione main
start:
    la a0, array         # a0 = indirizzo dell'array
    la a1, length        # a1 = lunghezza dell'array
    lw a1, 0(a1)
    jal ra, sommaArray
```

```
li a7, 93  
ecall
```

```
# syscall exit  
# termina il programma
```