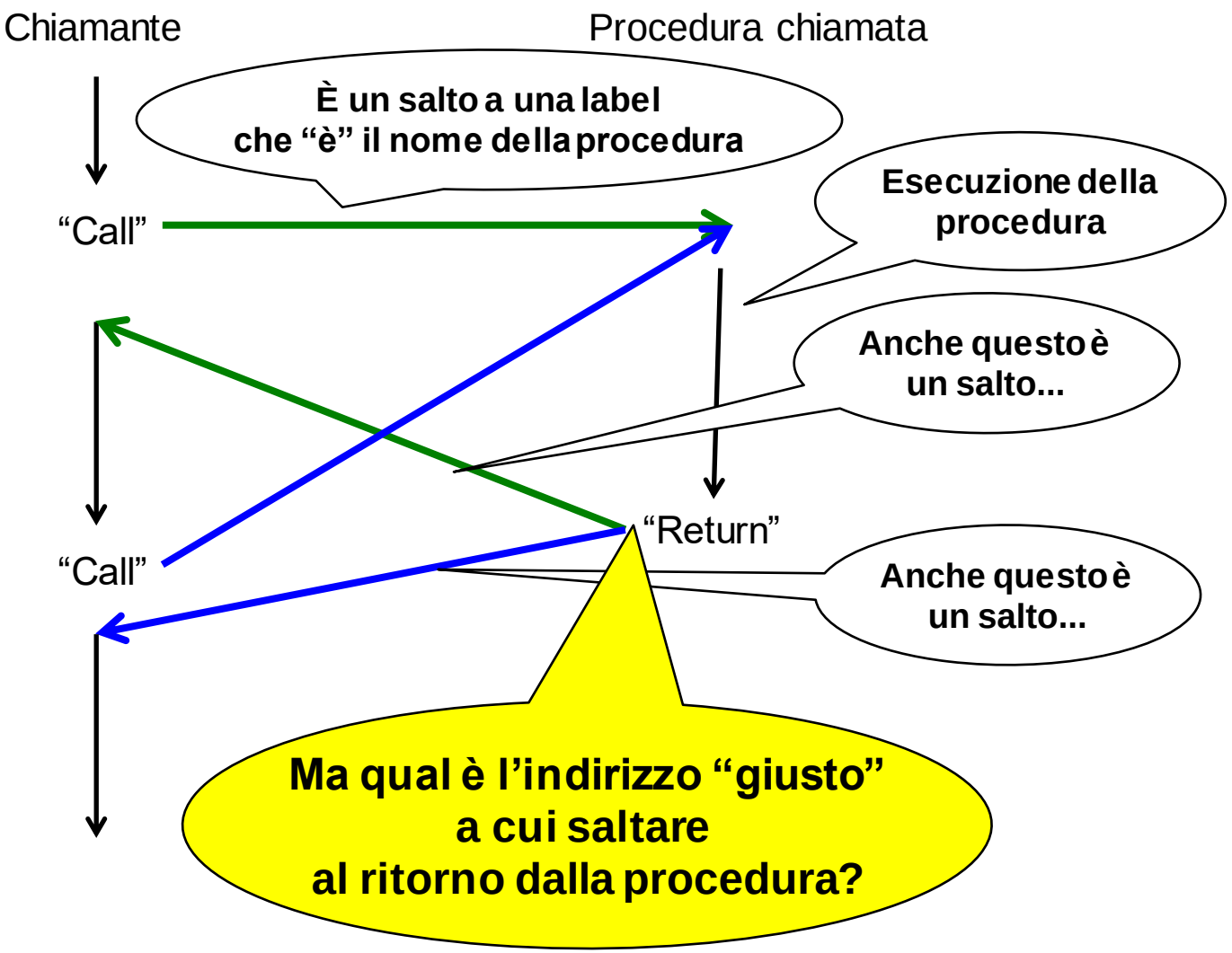
Corso di Laurea in Informatica
Architettura degli elaboratori
a.a. 2020-2021

Programmazione Assembly: procedure (1)

Passaggio parametri attraverso registri
Syscall

Claudia Raibulet
claudia.raibulet@unimib.it

Flusso di controllo



Istruzione jal

- **jal <IndirizzoProcedura>**
  - **(“jump and link”)**
  - **Salta a una procedura indicata nell’istruzione e contemporaneamente crea un collegamento a dove deve ritornare per continuare l’esecuzione del chiamante**
  - Salva nel registro $ra (registro 31) (“return address”) l’indirizzo a cui tornare dopo l’esecuzione della procedura
  - (è l’indirizzo successivo a quello dell’istruzione jal, cioè l’indirizzo in cui si trova la jal + 4)
  - Tale indirizzo si trova nel registro PC (Program Counter)

Istruzione jr

- **jr <registro>**
  - (“jump register”)
  - Salta all’indirizzo contenuto in un registro
  - È una istruzione **di uso generale** che consente di saltare a qualsiasi locazione di memoria, MA...
- **jr $ra**
  - Uno degli utilizzi tipici di jr
  - Per realizzare il ritorno da procedura
  - Saltando all’indirizzo precedentemente salvato da jal

Passaggio parametri (convenzioni base)

- **$a0 - $a3: registri argomento per il passaggio dei parametri**
- **$v0 - $v1: registri valore per la restituzione dei risultati**
- **NB: dal punto di vista hw sono registri come tutti gli altri, MA...**
- **...il loro utilizzo per il passaggio di parametri e risultati è una convenzione programmativa che deve essere rispettata per consentire di scrivere procedure che**
  - Possono essere scritte senza bisogno di sapere come è fatto il programma che le chiama
  - Possono essere chiamate senza bisogno di sapere come sono fatte "dentro"
- **NB: un parametro può essere un dato o un indirizzo!!!**
  - Rivedere passaggio parametri "per valore" o "per indirizzo" dall'insegnamento di Programmazione 1
  - Confrontare le istruzioni la e lw

  - la $s0, label

Osservazione

- **Una procedura – un meccanismo per organizzare in modo comprensibile e riutilizzabile il codice**
- **Le procedure consentono ai programmatori di concentrarsi su una parte del problema alla volta**

- **Comparazione tra procedure e spie: le spie lavorano in segreto, acquisiscono risorse, svolgono il compito, nascondono le tracce, e tornano al punto di partenza con i risultati richiesti**
- **I 6 passi di una procedura:**
  - Setting dei parametri in un luogo accessibile alla procedura
  - Trasferire il controllo alla procedura e salvare l'indirizzo dell'istruzione dove tornare dopo la chiamata della procedura (usare l'istruzione jal)
  - Acquisire risorse per l'esecuzione della procedura
  - Eseguire il compito richiesto
  - Mettere il risultato in un luogo accessibile al chiamante
  - Restituire il controllo al punto di partenza (usare l'istruzione jr)

Riferimenti principali

- **Patterson – Hennessy, Computer Organization and Design, Morgan Kaufmann**
  - Capitolo 2, Sezione 8
- **Appendice B (A nella vecchia edizione)**

Esempio elementare (1): dati

Il programma definisce quattro numeri num1, num2, num3 e num4.
Definisce una procedura “somma1” che riceve due numeri come parametri e restituisce la loro somma.
Chiama due volte la procedura passando come parametri prima num1 e num2 (memorizzando il risultato in result1), poi num3 e num4 (memorizzando il risultato in result2)

```
.data
num1:  .word 50
num2:  .word 14
result1: .word 0
num3:  .word 50
num4:  .word -66
result2: .word 0
```

Esempio elementare (2): programma principale

```
.text
.globl main
main:

#prima chiamata della procedura
la $t0, num1        # carica in $t0 un indirizzo (load address)
lw $a0, 0($t0)      #predisposizione del primo parametro $a0
la $t0, num2
lw $a1, 0($t0)      #predisposizione del secondo parametro $a1

jal somma1
#l'indirizzo della istruzione successiva viene salvato in $ra e si
# salta alla procedura.

sw $v0, result1     # memorizzazione del primo risultato
                    # la $t0, result1
                    # sw $v0, 0($t0)

#seconda chiamata della procedura
la $t0, num3
lw $a0, 0($t0)
la $t0, num4
lw $a1, 0($t0)
jal somma1
```

Esempio elementare (3): la procedura

```
#Procedura
somma1: #questo è l'indirizzo iniziale della procedura

add $v0, $a0, $a1
# convenzione:
# I registri $a0-$a3 si usano per passare i parametri
# I registri $v0 e $v1 si usano per restituire i risultati

jr $ra
# il registro $ra contiene l'indirizzo di ritorno
```

Problemi aperti

- **Cosa succede se una procedura ne chiama un'altra?**
  - Si perde il contenuto di $ra della prima chiamata?
  - Procedure recursive?
  - -> uso dello stack
- **Se una procedura usa registri, cosa succede del contenuto lasciato nei registri dal chiamante?**
  - Convenzioni: registri $s e $t
- **Dove stanno le variabili locali della procedura?**
  - Stack frame ...
- **Di tutto questo parleremo più avanti...**
- **...ma cominciate a porvi i problemi**