Catena programmativa

Esercitazione 6
Architettura degli elaboratori

Esercizio 1

Dato il seguente frammento di codice:

```
.data 0x10010000

ciao:
        .asciiz “Ciao Ciao.”
mamma:
        .ascii “Mamma”
t:
        .space 6
eol:
        .asciiz "1"
```

Quale valore assume l'etichetta ”t”?

Quale valore assume l'etichetta ”eol”?

Esercizio 1 – soluzione

- Dato il seguente frammento di codice:

```
.data 0x10010000

ciao:
        .asciiz “Ciao Ciao.”
mamma:
        .ascii “Mamma”
t:
        .space 6
eol:
        .asciiz "1"
```

Quale valore assume l'etichetta ”t”?

L’etichetta “t” assume valore 0x10010010, ovvero è l’indirizzo 10+1+5 (=16) dopo l’inizio del segmento data (che inizia dall’indirizzo 0x10010000).

Infatti la direttiva “.asciiz” aggiunge anche il carattere di fine stringa mentre la direttiva “.ascii” no

Quale valore assume l'etichetta ”eol”?

L’etichetta eol inizia 6 byte dopo l’etichetta t, ovvero all’indirizzo 0x10010016

Esercizio 2

- Cosa sono le pseudoistruzioni e come si comporta un assemblatore quando ne incontra una?

Esercizio 2 – soluzione

- Cosa sono le pseudoistruzioni e come si comporta un assemblatore quando ne incontra una?

- Le pseudoistruzioni sono istruzioni non implementate dall'hardware (cioè, non native dell'ISA di MIPS)
- Sono tradotte dall'assemblatore in sequenze di istruzioni native
- Secondo le convenzione d'uso dei registri, il registro $1 ($at) è riservato a questo scopo

Esercizio 4

Quali direttive assembly permettono di definire str in modo che possa essere visibile da altri moduli del programma?

In quale porzione delle memoria sarà salvata str?

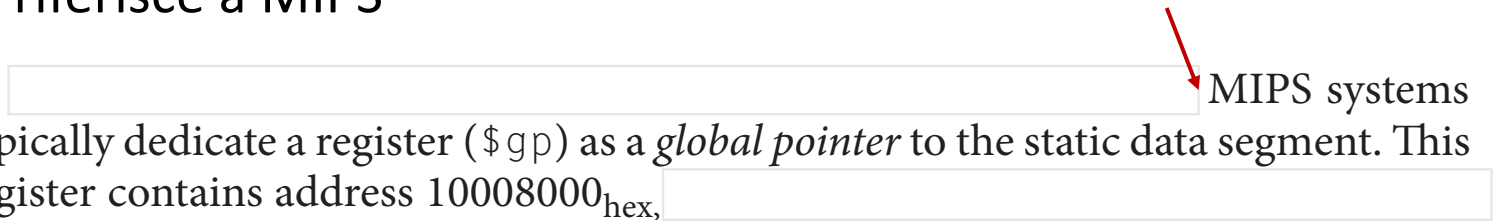
Esercizio 4 – soluzione

Le direttive assembly che permettono di definire str in modo che possa essere visibile da altri moduli del programma sono:

- .globl str #visibile all'esterno
- .extern str 16 #visibile all'esterno e di 16 byte

La porzione delle memoria in cui sarà salvata str è quella riservata ai dati statici

N.B. il simulatore QTSPIM potrebbe memorizzare i dati statici a partire da un indirizzo diverso (e.g. 0x10010000) da quello indicato dal libro di testo (pag. A-21) che si riferisce a MIPS



Esercizio 5

Dato il seguente programma (non ci sono istruzioni prima o dopo quelle qui elencate)

```
.data 0x10010000
.text
#qui ci sono 2 istruzioni, nessuna pseudoistruzione e non ci sono definizioni o uso di label
j primaLabel
secondaLabel:
#qui ci sono 3 istruzioni, nessuna pseudoistruzione e non ci sono definizioni o uso di label
terzaLabel:
#qui ci sono 4 istruzioni, nessuna pseudoistruzione e non ci sono definizioni o uso di label
j secondaLabel
```

A quante delle label presenti, l'assemblatore assocerà correttamente un indirizzo?
Quante e quali label saranno “unresolved” dopo l’assemblaggio?
Se al simbolo secondaLabel è associato il valore 0x00400030, a quale valore è associato il simbolo “terzaLabel”?

Esercizio 5 – soluzione

Dato il seguente programma (non ci sono istruzioni prima o dopo quelle qui elencate)

```
.data 0x10010000
.text
#qui ci sono 2 istruzioni, nessuna pseudoistruzione e non ci sono definizioni o uso di label
j primaLabel
secondaLabel:
#qui ci sono 3 istruzioni, nessuna pseudoistruzione e non ci sono definizioni o uso di label
terzaLabel:
#qui ci sono 4 istruzioni, nessuna pseudoistruzione e non ci sono definizioni o uso di label
j secondaLabel
```

L'assemblatore assocerà correttamente un indirizzo a 2 label, ovvero “secondaLabel” e “terzaLabel”

Sarà invece“unresolved” dopo l’assemblaggio l’etichetta “primaLabel” che viene usata ma non dichiarata

Al simbolo “terzaLabel” verrà associato il valore 0x00400030 + 12_{10} (=3*4) → 0x0040003c poichè si trova 3 istruzioni (ciascuna della dimensione di una word) dopo secondaLabel

Esercizio 6

- Qual è l'ordine corretto di esecuzione dei passi necessari per trasformare un programma scritto in un linguaggio di alto livello in un programma eseguibile?
  - Caricare, compilare, assemblare e linkare
  - Compilare, assemblare, linkare e caricare
  - Compilare, assemblare e linkare
  - Assemblare, compilare, linkare e caricare
- Qual è l'ordine corretto di esecuzione dei passi necessari per trasformare un programma scritto in un linguaggio di alto livello in un programma in esecuzione su un calcolatore?
  - Caricare, compilare, assemblare e linkare
  - Compilare, assemblare, linkare e caricare
  - Compilare, assemblare e linkare
  - Assemblare, compilare, linkare e caricare
- e per ottenere un file oggetto?
  - Caricare, compilare, assemblare e linkare
  - Compilare, assemblare e linkare
  - Compilare, assemblare, linkare e caricare
  - Compilare e assemblare

Esercizio 6 – soluzione

- Qual è l'ordine corretto di esecuzione dei passi necessari per trasformare un programma scritto in un linguaggio di alto livello in un programma eseguibile?
  - Caricare, compilare, assemblare e linkare
  - Compilare, assemblare, linkare e caricare
  - Compilare, assemblare e linkare
  - Assemblare, compilare, linkare e caricare
- Qual è l'ordine corretto di esecuzione dei passi necessari per trasformare un programma scritto in un linguaggio di alto livello in un programma in esecuzione su un calcolatore?
  - Caricare, compilare, assemblare e linkare
  - Compilare, assemblare, linkare e caricare
  - Compilare, assemblare e linkare
  - Assemblare, compilare, linkare e caricare
- e per ottenere un file oggetto?
  - Caricare, compilare, assemblare e linkare
  - Compilare, assemblare e linkare
  - Compilare, assemblare, linkare e caricare
  - Compilare e assemblare

Esercizio 7

- Quali informazioni sono output di un assemblatore?
- Quali informazioni sono output di un linker?

Esercizio 7 – soluzione

- L'output di un assemblatore è il file oggetto (**object file**), il cui contenuto è:
  - **object file header** – dimensione e posizione delle altri parti del file
  - **text segment** – codice macchina delle istruzioni che compongono il programma contenuto nel file (alcune potrebbe far riferimento a riferimenti non risolti - unresolved references)
  - **data segment** – rappresentanzione binaria dei dati contenuti nel file (potrebbero essere incompleti a causa di riferimenti a etichette/label che si trovano in altri file)
  - **relocation information** – instruzioni e dati che dipendono da riferimenti assoluti
  - **symbol table** – associa gli indirizzi delle etichette/label globali (visibili all'esterno del file) agli indirizzi e elenca i riferimenti non risolti (unresolved references)
  - **debugging information** – descrizione sintetica (usabile, ad esempio, da un programma di debugging)

- L'output di un linker è il programma eseguibile (**executable file**). Un programma eseguibile ha lo stesso formato del file oggetto MA *non contiene riferimenti non risolti e le relocation information*

Esercizio 8

- Due procedure (A e B) sono state assemblate separatamente e i file oggetto risultanti sono linkati mettendo A prima di B
- Sapendo che:
  - la dimensione del testo di A è di 0x200 e dei sui dati di 0x10
  - la dimensione del testo di B è di 0x100 e dei sui dati di 0x30

- Qual è l'indirizzo base a partire dal quale sarà allocato il testo di A nel file eseguibile?
- Qual è, nel file oggetto di A, il valore del campo address di una chiamata alla procedura B (jal B)?
- Qual è, nel file eseguibile, il valore del campo address di una chiamata in A alla procedura B?

Esercizio 8 – soluzione

- Due procedure (A e B) sono state assemblate separatamente e i file oggetto risultanti sono linkati mettendo A prima di B
- Sapendo che:
  - la dimensione del testo di A è di 0x200 e dei sui dati di 0x10
  - la dimensione del testo di B è di 0x100 e dei sui dati di 0x30

- Qual è l'indirizzo base a partire dal quale sarà allocato il testo di A nel file eseguibile?

0x0040 0000 (subito dopo area riservata – v. pag A-20)

- Qual è, nel file oggetto di A, il valore del campo address di una chiamata alla procedura B (jal B)?

indefinito, perchè sarà noto e inserito, solo quando A e B verranno uniti dal linker nell'eseguibile

- Qual è, nel file eseguibile, il valore del campo address di una chiamata in A alla procedura B?

Sapendo che B si trova dopo A e che A occupa uno spazio di 0x200, B inizierà all'indirizzo

0x0040 0200

Esercizio 9

- Cosa deve fare un assemblatore a due passi quando durante il primo passo:
  - incontra un'istruzione preceduta da un'etichetta?
    E.g. etichetta: add $t0, $t1, $t2
  - incontra un'istruzione non preceduta da un'etichetta in cui è presente l'uso di un simbolo (etichetta risolta)?
    E.g. j pippo
  - incontra un'istruzione senza etichetta in cui non sono presenti dei simboli?

Esercizio 9 – soluzione

- Cosa deve fare un assemblatore a due passi quando durante il primo passo:
  - incontra un'istruzione preceduta da un'etichetta?
    E.g. etichetta: add $t0, $t1, $t2

  scrive l'etichetta e il corrispondente valore (riferimento assoluto) nella tabella dei simboli
  - incontra un'istruzione non preceduta da un'etichetta in cui è presente l'uso di un simbolo (etichetta risolta)?
    E.g. j pippo

  cerca il simbolo nella tabella dei simboli e lo sostituisce con il valore corrispondente (se presente)
  - incontra un'istruzione senza etichetta in cui non sono presenti dei simboli?

  nulla

Esercizio 10

- Cosa è necessario fare per ottenere un programma eseguibile se le istruzioni che compongono il programma sono scritte in diversi file?

Esercizio 10 – soluzione

- Cosa è necessario fare per ottenere un programma eseguibile se le istruzioni che compongono il programma sono scritte in diversi file?

- Oltre ad assemblare i file separatamente, serve fare tutto ciò che fa il linker, cioè:
  - Ricercare le librerie usate dal programma (e.g. syscall)
  - Determinare (leggendolo dall'object header) la porzione di memoria che dovrà essere occupata dal codice di ogni modulo e riposizionare le istruzioni di ogni modulo sistemando gli eventuali indirizzi assoluti
  - Risolvere i riferimenti tra file
  - …

Esercizio 11

- Un riferimento non risolto (“unresolved reference”) può indicare un errore del programmatore. Quando?
  - Sempre
  - Se non è risolta dopo che è stato assemblato il file che contiene l’uso del simbolo
  - Se non è risolta dopo che sono stati assemblati tutti i file che compongono il programma
  - Se non è risolta dopo che il linker ha terminato la sua attività

Esercizio 11 – soluzione

- Un riferimento non risolto (“unresolved reference”) può indicare un errore del programmatore. Quando?
  - Sempre
  - Se non è risolta dopo che è stato assemblato il file che contiene l’uso del simbolo
  - Se non è risolta dopo che sono stati assemblati tutti i file che compongono il programma
  - **Se non è risolta dopo che il linker ha terminato la sua attività**

Esercizio 12

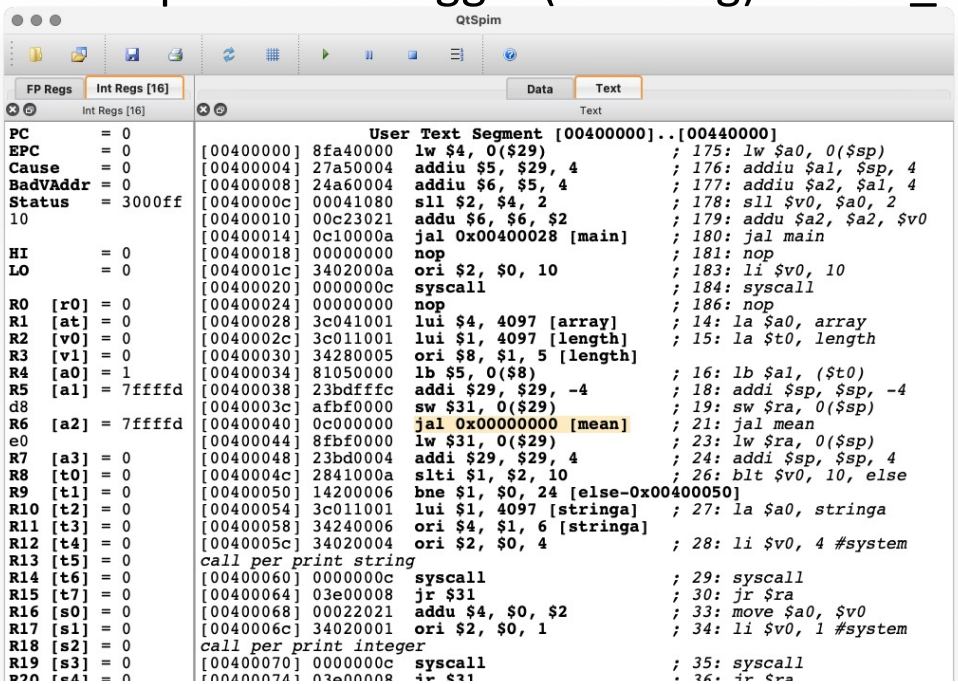
- Se un programma è scritto in due file, quando un simbolo dichiarato come global in un file risolve un riferimento utilizzato in un altro file?

Esercizio 12 – soluzione

- Se un programma è scritto in due file, quando un simbolo dichiarato come global in un file risolve un riferimento utilizzato in un altro file?
  - Quando entrambi fanno riferimento ad un'etichetta con lo stesso nome

- Esempio:
  - mean_lib.asm: …
  - vector_mean.asm: … usa mean (dichiarata dove? come?)
- Valore etichetta mean

… dopo assemblaggio (e linking) vector_mean.asm



… dopo assemblaggio mean_lib.asm