

# Lezione 3: L'architettura LC-3

## Laboratorio di Elementi di Architettura e Sistemi Operativi

22 Dicembre 2020

### RICORDO

#### Pseudo direttive assembly

- Per scrivere un programma in assembly sono necessarie alcune pseudo direttive (chiamate anche pseudo-ops);
- esse non sono vere e proprie istruzioni da eseguire, ma...

**servono in fase di traduzione da ASM a linguaggio macchina per far sì che il traduttore interpreti correttamente il contenuto del programma.**

- si riconoscono perché iniziano con il "." e, ovviamente, sono nomi riservati.
- Le pseudo-direttive del linguaggio assembly di LC-3 sono:

**.ORIG:** indica l'indirizzo di partenza del programma LC-3; ad esempio alla riga 05 viene detto che il programma deve iniziare all'indirizzo (esadecimale su 16 bit) x3050;

**.END:** indica dove finisce il programma. Attenzione: questa istruzione non dice che il programma deve essere terminato, ma l'indirizzo a cui terminano le sue istruzioni!

**.FILL:** indica che la prossima locazione di memoria contiene il valore indicato dall'operando;

**.BLKW:** indica di riservare una sequenza (contigua), lunga tanto quanto indicato dall'operando, di locazioni di memoria.  
Si possono anche inizializzare;

**.STRINGZ:** indica di iniziare una sequenza lunga n+1 di locazioni di memoria. L'argomento è una sequenza di n caratteri compresa fra i doppi apici.

I caratteri vengono rappresentati su 16bit.  
Il carattere terminatore '\0' è sottointeso.

# La gestione delle sub-routine

## ISTRUZIONE Jump Sub Routine ( JSR ):

### Jump Sub Routine Register (JSRR ):

- Le istruzioni **JSR** (codice operativo = 0100 ) Saltano all'indirizzo che segue (eventualmente contenuto nel registro (se usate JSRR)

L'istruzione inoltre salva in **R7** l'indirizzo dell'istruzione successiva alla JSR (JSRR)

- Va usata in coppia con l'istruzione **RET** che ripristina il **PC** al valore precedentemente salvato in **R7**

## ISTRUZIONE RET:

- L'istruzioni **RET** (codice operativo = 1100 ) ripristina nel **PC** il valore contenuto in **R7**, in pratica ritorna all'istruzione successiva alla **JSR**
- Esempio:** si provi a scrivere un subroutine che esegue la sottrazione tra R0 e R1 e il risultato lo pone in R2:  $R2 = R0 - R1$ ;

Ricordo che non esiste un'istruzione assembler che svolge la sottrazione, per poter svolgere una sottrazione, è necessario complementare (comabiare il segno) al registro con il sengo meno e poi svolgere la somma.

Scrivere il seguente programma, compilarlo e mandarlo in secuzione

```
.ORIG x3000
LD R0, VAR1
LD R1, VAR2
BRznp SOTTRAZIONE
JSR SOTTRAZIONE
ST R2, RISULTATO
HALT

SOTTRAZIONE:
NOT R1, R1
ADD R1, R1, #1
ADD R2, R0, R1
RET
; Definizione di variabili
VAR1 .FILL 0040 ; Valore espresso in decimale
VAR2 .FILL 0015

RISULTATO .FILL 0

.END
```

# La gestione delle TRAP

## ISTRUZIONE System CALL ( TRAP ):

- L'istruzione **TRAP** (codice operativo = 1111 ) Saltano all'indirizzo che è espresso come indice di un vettore precostituito di funzioni (si faccia riferimento alla tabella A.2 nel PDF pubblicato contenete l'elenco delle istruzioni).
- In linea di principio ci sono 256 funzioni precostituite da x0000 a 0x00FF
- Ad esempio **x22** è la funzione che stampa una stringa il cui indirizzo è contenuto in **R0** e la stringa deve essere terminata da \0

Scrivere il seguente programma, compilarlo e mandarlo in esecuzione

```
.ORIG x3000
    LEA    R0, STRINGA
    TRAP   X22 ; PUTS

    HALT

; Definizione di variabili
STRINGA    .STRINGZ    "Ciao Mondo" ; Stringa già terminata con \0

.END
```

Esercizi si scrivano in assembler LC3 i seguenti programmi:

**Esercizio 1:**

Si scriva un programma assembler che alla pressione di una lettera maiuscola sulla tastiera stampi la lettera minuscola corrispondente.  
Il programma termina alla pressione del tasto 7

**Soluzione 1:**

```
.ORIG X3000
    LD R2, TERM      ; Load -7 lo usiamo come test per l'uscita
    LD R3, ASCII      ; Load ASCII differenza tra maiuscole e minuscole
AGAIN:
    TRAP x23          ; Legge un carattere da tastiera
    ADD R1, R2, R0     ; Se vale 7 (sommo -7 e va a zero)
    BRz EXIT          ; salta alla fine
    ADD R0, R0, R3     ; sommo per arrivare ai caratteri minuscoli
    TRAP x21           ; stamp il carattere a video
    BRnzp AGAIN        ; ripeto...

    ; Differenza tra maiuscole e minuscole
EXIT:    TRAP x25      ; Halt
.END
```

**Esercizio 2:**

Si scriva un programma assembler che prenda in input 2 valori numerici, li sommi e stampi a video il risultato.

**Esercizio 3:**

Si scriva un programma assembler che prenda in input una stringa, e una lettera e calcoli quante volte appare la lettera nella stringa.

**Esercizio 4:**

Si modifichi l'esercizio 2 e lo si trasformi in una funzione