Report S10-L3

Nella lezione teorica del mattino, abbiamo visto i fondamenti del linguaggio Assembly. Dato il codice in Assembly per la CPU x86 allegato qui di seguito, identificare lo scopo di ogni istruzione, inserendo una descrizione per ogni riga di codice. Ricordate che i numeri nel formato 0xYY sono numeri esadecimali. Per convertirli in numeri decimali utilizzate pure un convertitore online, oppure la calcolatrice del vostro computer (per programmatori).

```
0x00001141 <+8>:     mov    EAX,0x20
0x00001148 <+15>:    mov    EDX,0x38
0x00001155 <+28>:    add    EAX,EDX
0x00001157 <+30>:    mov   EBP, EAX
0x0000115a <+33>:    cmp   EBP,0xa
0x0000115e <+37>:    jge  0x1176 <main+61>
0x0000116a <+49>:     mov  eax,0x0
0x0000116f <+54>:    call  0x1030 <printf@plt>
```

I registri EAX, EBX, ECX, EDX, ESI, EDI, EBP sono registri “general purpose” a 32 bit. Servono come sorgente o destinazione per qualunque operazione di trasferimento e aritmetico logica.

Sebbene i registri “general purpose” possano essere usati per qualsiasi scopo, esistono delle istruzioni che li usano in modo specializzato rendendo molto efficiente il loro utilizzo. La sigla del registro spesso deriva proprio da questo uso specializzato

- EAX: Accumulatore dei risultati, sebbene qualsiasi registro general purpose possa essere usato per questo scopo è preferibile usare EAX per accumulare risultati di operazioni aritmetico/logiche
- EBX: Puntatore alla memoria dei dati, anche in questo caso qualunque registro general purpose a 32 bit è in grado di puntare ad una cella di memoria ma EBX è la scelta preferenziale
- ECX: Contatore di iterazioni, qualunque registro può essere usato come contatore in un algoritmo iterativo ma esistono istruzioni specializzate che usano ECX come contatore
- DX: Puntatore alle porte di I/O: le istruzioni di I/O sono fortemente specializzate e solo il registro a 16 bit DX può essere usato come puntatore ad una porta di I/O (spazio di I/O =64K porte)
- ESI: Puntatore alla memoria dei dati sorgente nelle operazioni di copia di stringa
- EDI: Puntatore alla memoria dei dati destinazione nelle operazioni di copia di stringa
- EBP: Puntatore ai parametri del sottoprogramma nello stack

0x00001141 <+8>: mov EAX,0x20
0x00001148 <+15>: mov EDX,0x38

Istruzione Move, copia
Carica il valore esadecimale 20 (decimale 32) nel registro EAX
Carica il valore esadecimale 38 (decimale 56) nel registro EDX

0x00001155 <+28>: add EAX,EDX

Istruzione add
Somma i valori contenuti nei registri EAX ed EDX, e salva il risultato nel registro EAX
Quindi EAX ora contiene 32 + 56 = 88

0x00001157 <+30>: mov EBP, EAX

Istruzione Move, copia
Copia il valore contenuto nel registro EAX (che è 88) nel registro EBP

0x0000115a <+33>: cmp EBP,0xa

Istruzione Compare
Confronta il valore nel registro EBP (88) con il valore esadecimale 0xa (decimale 10)

A seguito di questa sottrazione, vengono impostati diversi flag all'interno del processore, in particolare i flag:

- ZF (Zero Flag): Viene impostato a 1 se il risultato della sottrazione è zero (cioè se EBP è uguale a 10).
- SF (Sign Flag): Viene impostato a 1 se il risultato della sottrazione è negativo (cioè se EBP è minore di 10).
- CF (Carry Flag): Viene impostato a 1 se si è verificato un overflow con riporto durante la sottrazione
- OF (Overflow Flag): Viene impostato a 1 se si è verificato un overflow complemento a due durante la sottrazione

0x0000115e <+37>: jge 0x1176 <main+61>

Istruzione Jump if Greater or Equal, salto condizionale
Se il valore in EBP è maggiore o uguale a 10, salta all'indirizzo di memoria 0x1176 (che è l'etichetta "main+61")
Altrimenti, continua con l'istruzione successiva ovvero

0x0000116a <+49>: mov EAX,0x0

Istruzione Move, copia
Se il confronto precedente è stato falso (EBP < 10), carica il valore 0 nel registro EAX

0x0000116f <+54>: call 0x1030 <printf@plt>

Istruzione call
Effettua una chiamata alla funzione printf (la cui libreria è indicata da @plt) passando come argomento il valore contenuto nel registro EAX

Confronto con MIPS

```
mov EAX,0x20
mov EDX,0x38

add EAX,EDX
mov EBP, EAX

cmp EBP,0xa

jge 0x1176 <main+61>


mov eax,0x0
call 0x1030 <printf@plt>
```

```
.data
str: .asciiz "Il risultato è minore di 10"

.text

main:
    li $t0, 20                  #carico 20 in t0
    li $t1, 38                  #carico 38 in t1

    add $t0, $t0, $t1           #somma in t0
    move $s0, $t0               #copio il contenuto di t0 in s0

    blt $s0, 10, stampa         #se s0<10 allora salto a stampa

    j end                       # Jump termine programma

stampa:
    la $a0, str                 # carico la stringa in a0
    li $v0, 4                   #system call per printf
    syscall

end:
    li $v0, 10                   #system call per exit
    syscall
```