ESERCITAZIONE WEEK 22 DAY 2

Traccia:

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).

```
0×00001141 <+8>:    mov   EAX,0×20
0×00001148 <+15>:   mov   EDX,0×38
0×00001155 <+28>:   add   EAX,EDX
0×00001157 <+30>:   mov   EBP, EAX
0×0000115a <+33>:   cmp   EBP,0xa
0×0000115e <+37>:   jge    0×1176 <main+61>
0×0000116a <+49>:   mov   eax,0×0
0×0000116f <+54>:   call   0×1030 <printf@plt>
```

0x00001141 <+8>: mov EAX,0x20	Assegna il valore **32** al registro EAX. Questo è un esempio di un'operazione di assegnamento in linguaggio assembly x86. Si può dire che viene copiato il valore 32 nel registro EAX, ovvero quello di solito usato per operazioni aritmetiche, memorizzazione del risultato e passaggio di argomenti.
0x00001148 <+15>: mov EDX,0x38	Assegna il valore **56** al registro EDX, usato per operazioni aritmetiche e per memorizzare la parte alta del risultato di moltiplicazioni e divisioni.
0x00001155 <+28>: add EAX,EDX	Somma il contenuto del registro EDX al contenuto del registro EAX e memorizza il risultato nella destinazione, che è il registro EAX. EAX sarà 32+56 =**88**.
0x00001157 <+30>: mov EBP, EAX	Copia il contenuto del registro EAX nel registro EBP, punto di riferimento per l'accesso ai parametri e alle variabili locali di una funzione, **88**.
0x0000115a <+33>: cmp EBP,0xa	Confronta il valore contenuto nel registro EBP con il valore immediato 0xA (o **10** in decimale). Questa istruzione effettua una sottrazione implicita tra i due operandi, ma non memorizza il risultato. La differenza tra i due operandi viene utilizzata per impostare i flag di stato del processore, che possono essere utilizzati successivamente per prendere decisioni condizionali nel flusso del programma. **ZF sarà 0 e CF sarà 0**.
0x0000115e <+37>: jge 0x1176 <main+61>	L'istruzione "jge" è un'istruzione di salto condizionale che significa "jump if greater than or equal" (salta se maggiore o uguale). In questo caso, "jge 0x1176 <main+61>" indica che il flusso del programma **salterà all'indirizzo 0x1176 (che corrisponde a un'istruzione specifica all'interno della funzione main, con un'offset di 61 rispetto all'inizio della funzione main**) se il confronto precedente (eseguito dall'istruzione "cmp EBP, 0xa") ha indicato che il valore contenuto

	nel registro EBP è maggiore o uguale a 10, condizione **vera: 88 > 10**.
0x0000116a <+49>: mov eax,0x0	Assegna il valore immediato (costante) 0x0 (o 0 in decimale) al registro EAX.
0x0000116f <+54>: call 0x1030 <printf@plt>	L'istruzione "call 0x1030 printf@plt" **chiama la funzione printf del programma**. L'indirizzo **0x1030 corrisponde all'indirizzo di memoria in cui è presente l'istruzione che invoca la funzione printf**. La parte <printf@plt> fa riferimento alla Global Offset Table (GOT) che viene utilizzata per l'indirizzamento dinamico delle funzioni in un programma compilato con **supporto per la chiamata a funzione dinamica**. In sostanza, questa istruzione chiama la funzione printf che sarà responsabile di stampare un output sulla console.

Registro	Nome	Funzione	General Purpose?
EAX	Accumulator Register	Utilizzato per operazioni aritmetiche, memorizzazione del risultato e passaggio di argomenti.	SÃ¬
EBX	Base Register	Utilizzato come registro base per accessi ai dati nella memoria.	SÃ¬
ECX	Count Register	Utilizzato come registro di conteggio in cicli e istruzioni di stringa.	SÃ¬
EDX	Data Register	Utilizzato per operazioni aritmetiche, memorizzazione della parte alta del risultato di moltiplicazioni e divisioni.	SÃ¬
ESI	Source Index	Utilizzato come indice nella copia dei dati da una posizione di memoria ad un'altra.	SÃ¬
EDI	Destination Index	Utilizzato come indice nella scrittura dei dati in una posizione di memoria.	SÃ¬
ESP	Stack Pointer	Punto di riferimento per lo stack. Utilizzato per indicare l'indirizzo del prossimo elemento nello stack.	No
EBP	Base Pointer	Punto di riferimento per l'accesso ai parametri e alle variabili locali di una funzione.	No
EIP	Instruction Pointer	Indica l'indirizzo dell'istruzione corrente in esecuzione. Utilizzato per eseguire il flusso di esecuzione del programma.	No
EFLAGS	Flags Register	Registro dei flag che indica il risultato delle operazioni aritmetiche e logiche.	No

ESERCITAZIONE WEEK 22 DAY 3

EPICODE

Esercizio
Linguaggio Assembly

Traccia:
Scrivere un programma in linguaggio assemblativo 8088 che, presi due dati a e b in memoria, calcola l'espressione (a+3)*b ponendo il risultato nel registro accumulatore.

a dd 12

b dd 5

In assembly x86 sarebbe:

mov EAX , [ind_a]

add EAX, 3

mov EBX, [ind_b]

mul EBX -> moltipico il valore di EBX a EAX

Ma in Assembly 8088 (16 bit), il registro accumulatore è detto AX:

mov AX , [ind_a]

add AX, 3

mov BX, [ind_b]

mul BX -> moltipico il valore di BX a AX