L’esercizio di oggi ha come traccia quanto segue:

“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>
```

Vediamo di seguito la **Soluzione**:

Possiamo dire, anzitutto che le istruzioni fornite rappresentano dei dump di codice assembly.

Ogni istruzione è divisa in due parti:

La **prima parte**, che rappresenta l’indirizzo di memoria a cui punta la funzione (che parte da 0x0 fino ai due punti). Per esempio della prima riga l’area a cui punta la funzione è: “**0x00001141 <+8>:”**, dove il numero in esadecimale è l’area di memoria dove viene memorizzato un dato o un istruzione, mentre il “<+8>” indica che l’istruzione si trova ad 8 byte dall’inizio della funzione.

La **seconda parte** rappresenta il tipo di funzione (nel caso della prima abbiamo “mov”), la destinazione(EAX) e l’argomento(0x20).

Da notare come sia l’indirizzo di memoria che l’argomento sono scritti in esadecimale; questo perché sono semplificazione della rappresentazione di un numero intero o decimale che la macchina ha memorizzato in linguaggio macchina(binario).

Sempre facendo riferimento alla **prima istruzione** vediamo il **funzionamento** completo del processo: si tratta di una funzione che sposta il valore “0x20” (che tradotto in decimale sarebbe “32”) all’interno del registro “EAX”. Questa istruzione è memorizzata nell’area “0x00001141” (che tradotto in decimale sarebbe l’area numero “4417”) della CPU.

L’istruzione è stata memorizzata ad una distanza di 8 byte dall’inizio dell’area numero 4417 della CPU.

Di seguito vediamo la **soluzione di tutte le istruzioni**:

0x00001141 <+8>: mov EAX,0x20 Sposta il valore intero decimale 32 nel registro EAX

0x00001148 <+15>: mov EDX,0x38 Sposta il valore intero decimale 56 nel registro EDX

0x00001155 <+28>: add EAX,EDX Somma il registro EDX ad EAX, praticamente somma 56 a 32 ed aggiorna il registro EAX con la somma = 88

0x00001157 <+30>: mov EBP, EAX Muove il contenuto del registro EAX, ovvero 88 nel registro EBP

0x0000115a <+33>: cmp EBP,0xa Controlla (CMP= compare) l'uguaglianza tra il valore 0xa che in decimale è il numero 10 con il valore contenuto in EBP ovvero 88

0x0000115e <+37>: jge 0x1176 <main+61> Effettua un salto condizionale se la destinazione di «cmp» è maggiore o uguale del valore di controllo. Considerato che 88 > 10, il salto viene effettuato

0x0000116a <+49>: mov eax,0x0 Sovrascrive il valore di EAX con il valore 0, ovvero sposta 0 in EAX

0x0000116f <+54>: call 0x1030 <printf@plt> Chiamata di funzione ad una funzione a noi nota, è la funzione printf