**ASSEMBLY X86**

**TRACCIA**identificare lo scopo di ogni istruzione, inserendo una descrizione per ogni riga di codice. Ricordate che i numeri nel formato 0xYY sono numeri esadecimali.  
  
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>  
  
**ESERCIZIO**  
L'etichetta “ 0x00001141 <+8> ” si riferisce all'offset rispetto all'indirizzo di memoria di partenza della funzione o del blocco di codice.   
Questo offset indica la posizione relativa all'inizio della funzione o del blocco di codice in esame.  
In questo contesto specifico, il valore +8 indica che l'istruzione corrispondente si trova a 8 byte dall'inizio della funzione. Pertanto, l'indirizzo effettivo dell'istruzione è ottenuto aggiungendo 8 byte all'indirizzo base della funzione.  
Vuol dire che se l'indirizzo di partenza della funzione è 0x00001141, a +8 sarebbe situata a 0x00001149.  
  
1) mov EAX,0x20

Convertiamo i numeri esadecimali in decimali.  
0x20 = 32 in decimale.  
L’istruzione carica il valore esadecimale 0x20 nel registro EAX.   
Ci ritroveremo, quindi, il valore 32 in EAX.  
  
2) mov EDX, 0x38  
0x38 = 56 in decimale.  
L’istruzione carica il valore esadecimale 0x38 nel registro EDX.   
Ci ritroveremo, quindi, il valore 56 in EDX.

3) add EAX, EDX  
L’istruzione somma il contenuto di EDX al registro EAX.   
Ci ritroveremo, quindi, il valore 88, cioè la somma di 32+56, in EAX.  
  
4) mov EBP, EAX  
L’istruzione carica il valore contenuto nel registro EAX nel registro EBP.   
Ci ritroveremo, quindi, il valore 88 in EBP.  
  
5) cmp EBP, 0xa   
0x38 = 56 in decimale.  
L’istruzione compara il valore contenuto nel registro EBP con il valore esadecimale 0xa.  
Viene fatta una sottrazione tra Destinazione e Sorgente (quindi 88-10) senza andare a modificare i valori iniziali.  
In base al risultato della sottrazione, verrano settati i flag ZeroFlag e CarryFlag utili alla prossima istruzione.  
Se:  
- Dest=Sorg, ZF=1 (il risultato è zero) e CF=0 (non vi è nessun riporto).  
- Dest<Sorg, ZF=0 e CF=1 (in quanto il risultato negativo è visto come riporto).  
- Dest>Sorg, ZF=0 e CF=0.  
  
6) jge 0x1176 <main+61>  
L’istruzione salta all'indirizzo 0x1176 se il risultato del cmp precedente restituisce i flag ZF e CF entrambi settati a 0. Ciò indica che la destinazione EBP è maggiore della sorgente 0xa.  
In questo caso la funzione jge ha successo e ci porterà all’indirizzo 0x1176 che si trova 61 byte più avanti rispetto all’indirizzo in cui ci troviamo.  
  
**Nel caso di jge fallito**7) mov EAX, 0x0  
0x0 = 0 in decimale.  
Se il cmp precedente è falso (cioè il valore in EBP<0xa), il valore 0 verrà caricato in EAX.

8) call 0x1030 <printf@plt>  
L’istruzione chiama la funzione printf, per stampare qualcosa a schermo. L'indirizzo 0x1030 potrebbe rappresentare l'indirizzo della funzione printf nella tabella delle procedure di collegamento dinamico (PLT).