DESCRIZIONE CODICE IN ASSEMBLY

L'esercizio di oggi ci chiede di descrivere brevemente la funzione di ogni riga del codice sottostante scritto in Assembly:

```
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>
```

mov EAX, 0x20: mov sposta la variabile all'interno del registro EAX; 0x20 è scritto in esadecimale e, utilizzando un convertitore, si traduce in decimale come 32.

mov EDX, 0x38: come prima, mov sposta la variabile nel registro EDX; 0x38 si traduce in 56 decimale. Possiamo dire che con queste prime 2 righe abbiamo dichiarato 2 variabili che sono 32 e 56.

add EAX, EDX: questo semplicemente fa l'addizione delle 2 variabili all'interno dei registri EAX e EDX, cioè 32+56 e salva il risultato sul primo registro, quindi EAX.

mov EBP, EAX: sposta il contenuto di EAX all'interno del registro EBP, che serve per tenere sotto controllo il flusso e l'esecuzione della funzione.

cmp EBP, 0xa: l'istruzione cmp funziona come una sottrazione ma non modifica gli operandi; in questo caso 0xa, 10 in decimale, viene sottratto a EBP che è 88 e viene fatto per capire se la destinazione (EBP) è uguale, maggiore o minore alla sorgente (0xa), tuto quesot servirà per la riga di codice sottostante.

jge 0x1176 <main+61>: jge sta a significare jump greater equal, cioè se la destinazione (EBP) è maggiore o uguale alla sorgente(0xa) salta alla locazione di memoria 0x1176.

mov EAX, 0x0: sposta 0x0, cioè zero in decimale, all'interno del registro EAX.

call 0x1030 <printf@plt>: chiama la funzione allocata nella parte di memoria 0x1030.