# ASSEMBLY x86

Dato il cosice in Assembly per CPUx86 allegato qui di seguito, identificare lo scopo di ogni istruzione, inserendo una descrizione per ogni riga.

### Traccia

```
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 <print@plt>
```

### Descrizione delle istruzioni:

**`Mov EAX,0x20`** = copia il valore della sorgente (0x20), in questo caso un esadecimale nel registro EAX

**`Mov EDX,0x38`** = copia il valore della sorgente (0x38), in questo caso un esadecimale nel registro EDX

**`Add EAX,EDX`** = somma il valore della sorgente (EDX) con quello del destinatario EAX

**`Mov EBP,EAX`** = copia il valore della sorgente (EAX) nel registro destinatario EBP

**`Cmp EBP,0xa`** = mette a confronto destinatario e sorgente per restituire un valore binario (ZF 0 – CF 0)

**`Jge 0x1176 <main+61>`** = Salta alla locazione di memoria 0x1176 se la destinazione è > = della sorgente “cmp”

**`Mov EAX,0x0`** = copia il valore della sorgente (0x0), in questo caso un esadecimale nel registro EAX

**`Call 0x1030 <print@plt>`** = con l’istruzione **call** viene chiamata una funzione. L’istruzione **call** passa l’esecuzione del programma alla funzione chiamata (creando un nuovo stack).

Per la conversione dei numeri esadecimali in numeri decimali si può usare il tool online qui di seguito:
https://www.rapidtables.org/it/convert/number/hex-to-decimal.html

A = 32

C = 56

somma = A+C (88)

E = A(88)

E(88) -0xa(10) - if E(88) ><= 0xa(10) – (ZF 0 – CF 0)

then E >= 0xa – condizione verificata avviene il salto e forse l’istruzione

A=0

call 0x1030 <print@plt>