CONSEGNA S10-L3

basic assembly

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>
```

questa sezione indica l'indirizzo di memoria nella tabella relazionale in cui ogni riga di questo codice assembly x86 si trova

0×00001141 <+8>:

0×00001148 <+15>:

0×00001155 <+28>:

0×00001157 <+30>:

0×0000115a <+33>:

0×0000115e <+37>:

0×0000116a <+49>:

0×0000116f <+54>:

mov EAX,0×20 mov EDX,0×38 add EAX,EDX

mov EBP, EAX

cmp EBP,0xa

jge 0×1176 <main+61>

mov eax,0×0

call 0×1030 <printf@plt>

questo e' il primo comando che verra' eseguito. mov e' l'istruzione e in questo caso viene utilizzato per indicare che al registro EAX deve essere assegnato il numero esadecimale 0x020 ovvero 32 in decimale

Questa e' la stessa azione di prima solo che definisce il valore da assegnare a EDX e il valore e' 0x38 ovvero 56 in decimale

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

call

0×0000116f <+54>:

0×1030 <printf@plt>

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>

con questa riga il programma aggiunge (add) il valore dei registri EAX a EDX ossia 32 + 56 ovvero 88 ed il risultato viene memorizzato sul registro EAX

questa riga col comando mov indica al registro EBP verra' assegnato il valore di EAX

```
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>

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

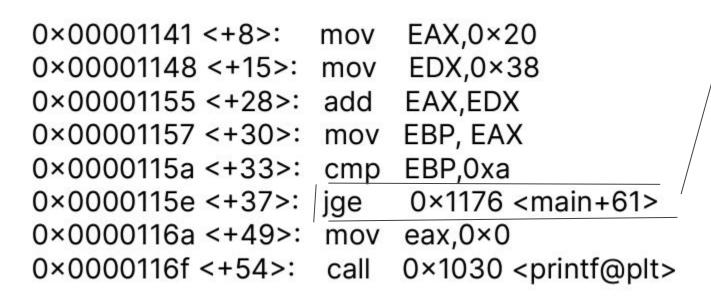
0×0000115a <+33>: <u>cmp</u> EBP,0xa

0×0000115e <+37>: jge 0×1176 <main+61>

0×0000116a <+49>: mov eax,0×0

0×0000116f <+54>: call 0×1030 <printf@plt>

il comando cmp in questa riga confronta il valore di EBP con il valore 10. Dopo cio' i flag del registro saranno impostando eseguendo EBP - 10 Se l'istruzione cmp ha confrontato EBP con il valore 10 e ha stabilito che EBP è maggiore o uguale a 10, l'istruzione jge 0x1176 salterà alla locazione di memoria 0x1176. Altrimenti, il flusso del programma continuerà normalmente con l'istruzione successiva.



con l'istruzione jge
0x1176 si saltera'
alla memoria
0x1176 altrimenti il
flusso del
programma
continuera
normalmente

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

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

con call si richiama il comando printf