REPORT 11/01/2023

FACCILONGO DOMENICO

ASSEMBLY x86

TASK

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).

```
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>
```

ANALISI E VALUTAZIONE

Come da task andiamo ad analizzare ciascun istruzione andandola a descrivere e identificare cosa il codice esegue.

mov EAX, 0x20

- COMPONENTI:
  - mov = operazione che consente di copiare una variabile da una locazione ad un'altra, e che quindi viene utilizzata per leggere e scrivere in memoria.
  - EAX = operando, il quale in questo caso è un registro di destinazione messo a disposizione dai processori x86 a 32bit.
  - 0x20 = operando, valore immediato scritto in formato esadecimale (nella forma 0xYY).

- OPERAZIONE:
  - in tale istruzione come possiamo notare troviamo un operazione con sintassi ***mov*** dal quale copia il valore 0x20, che trasformeremo in valore decimale, nel registro EAX. Il valore 0x20 trasformato in decimale diventa *32*, che verrà quindi copiato nel registro EAX.

- EQUIVALENTE IN LINGUAGGIO C:
    - Se volessimo leggere l'istruzione assembly in linguaggio C diremo che il risultato sarà pressocchè questo:

```
int main () {
int a=32;
…
}
```

Ovvero che verrà dunque inizializzata una nuova variabile dal nome a (corrispondente al registro EAX) con suo valore pari a 32.

mov EDX, 0x38

- COMPONENTI:
    - Stesse dell'istruzione precedente
- OPERAZIONE:
    - L'operazione è uguale all'istruzione precedente, ovvero di copiare l'operando sorgente, nel registro destinazione, con la differenza che qui abbiamo come destinazione un registro differente sempre a 32 bit (EDX); mentre come operando sorgente abbiamo un nuovo numero esadecimale (0x38) che trasformeremo in decimale. Quindi 0x38 in decimale diventa 56 il quale verrà copiato nel nuovo registro EDX.
- EQUIVALENTE IN LINGUAGGIO C:
    - Anche in questo caso, in linguaggio C, proseguendo il codice C scritto precedentemente, l'istruzione diventa:

```
…
  int b = 56;
…
```

In cui è stata inizializzata una nuova variabile dal nome b (corrispondente al registro EDX con valore pari a 56.

add EAX, EDX

- COMPONENTI:
  - add = operazione che consente di sommare una variabile/registro ad un'altra, aggiornando/salvando il risultato nel registro/valore di destinazione.
  - EAX = operando, il quale in questo caso è un registro di destinazione messo a disposizione dai processori x86 a 32bit.
  - EDX = operando, il quale in questo caso è un registro di sorgente messo a disposizione dai processori x86 a 32bit.
- OPERAZIONE:
  - L'operazione consente di sommare il contenuto del registro EDX con il contenuto del registro EAX andando a salvare ed aggiornare il risultato nel registro di destinazione ovvero EAX. In tal caso avremo 32+56=88, il quale verrà inserito aggiornando il registro di destinazione (EAX).
- EQUIVALENTE IN LINGUAGGIO C:
  - in linguaggio C, proseguendo il codice C scritto precedentemente, l'istruzione diventa:

    …
    somma = a + b;
    a = somma
    …

    In cui è stata inizializzata una nuova variabile dal nome b (corrispondente al registro EDX con valore pari a 56.

mov EBP, EAX

- COMPONENTI:
  - Stesse dell'istruzione mov precedente con la differenza che abbiamo come operando di destinazione un registro a 32 bit EBP.
- OPERAZIONE:
  - L'operazione è uguale all'istruzione mov precedente, ovvero di copiare il contenuto del registro sorgente, nel registro destinazione. Quindi il contenuto del registro EBP diventa 88.

- EQUIVALENTE IN LINGUAGGIO C:
    - proseguendo il codice C scritto precedentemente, l'istruzione diventa:

…
Int c = a;
…

In cui è stata inizializzata una nuova variabile dal nome c (corrispondente al registro EBP con valore pari a 88.

cmp EBP, 0xa

- COMPONENTI:
    - cmp = istruzione che modifica i flag ZF (zero flag) e CF (carry flag)
- OPERAZIONE:
    - L'operazione consiste nell'andare a guardare "destinazione e sorgente" per poi modificare i due flag, e successivamente andare a sottrarre alla destinazione la sorgente. In questo caso trasformiamo 0xa in decimale e diventa 10 il quale verrà confrontato con il contenuto del registro EBP (= 88), ottenuti i flag in base alla comparazione dei due valori e infine sottratti. Avendo **destinazione>sorgente** la ZF e CF sono entrambi 0. Mentre il risultato dell'operazione di sottrazzione è 88-10=78. Quindi il valore del registro EBP è 78.
- EQUIVALENTE IN LINGUAGGIO C:
    - proseguendo il codice C scritto precedentemente, l'istruzione diventa:

…
int d = 10;
if (c => 10);
istruzione del jge
else
…

In cui è stata inizializzata una nuova variabile dal nome d (corrispondente al valore 10). Il cmp funge come un "if" in cui comparando due valori determina un istruzione.

jge 0x1176 <main+61>

- COMPONENTI:
  - jge = jump, inteso come salto condizionale che avviene se la destinazione è maggiore o uguale alla sorgente nell'istruzione cmp.
  - 0x1176= locazione di memori.
- OPERAZIONE:
  - In questo caso, il jump, si ricollega all'istruzione precedente (cmp), ponendo la condizione, che se si verifica che la destinazione è maggiore della sorgente allora l'operazione permette di saltare alla locazione indicata dal codice (0x1176)
- EQUIVALENTE IN LINGUAGGIO C:
  - proseguendo il codice C scritto precedentemente, l'istruzione diventa:

…
(dopo l'"if")
Istruzione nella locazione 0x1176
…

In cui dopo che la condizione if del cmp verrà verificata allora verrà eseguita l'istruzione della locazione in cui si è jumpato (jge).

mov EAX, 0x0

- COMPONENTI:
  - Stesse delle istruzioni precedenti
- OPERAZIONE:
  - L'operazione è uguale delle istruzioni precedenti.
- EQUIVALENTE IN LINGUAGGIO C:
  - In linguaggio C, proseguendo il codice scritto precedentemente, l'istruzione diventa:

…
return
int a = 0;
…

Questa è la condizione per cui il cmp non è verificato e che pertanto il ciclo "if" finisce nell'else che va a ricominciare il ciclo fino a quando la condizione non è verificata.

call 0x1030 <printf@plt>

- COMPONENTI:
  - call = L'istruzione avvia l'esecuzione del sottoprogramma (chiamata) ed è simile a un'istruzione di salto (jp) in quanto provoca un salto nella sequenza di esecuzione del programma.
  - 0x1030 = locazione di memoria.
- OPERAZIONE:
  - Call permette di saltare alla locazione indicata dal codice (0x1030).
- EQUIVALENTE IN LINGUAGGIO C:
  - il codice scritto precedentemente, l'istruzione diventa:

    …
    Istruzione della locazione 0x1030
    …

    L'istruzione manda il proseguirsi del codice all'istruzione della locazione di memoria indicata n. 0x1030.