Prendiamo in considerazione il seguente codice e andiamo a commentare e descrivere ogni singola istruzione:

```
MOV EAX, 0x20
MOV EDX, 0x38
ADD EAX, EDX
MOV EBP, EAX
CMP EBP, 0xa
JGE 0x1176 <main+61>
MOV EAX, 0x0
CALL 0x1030 <printf@plt>
```

Prima di iniziare a commentare andiamo a convertire i numeri esadecimali in numeri decimali per ottenere:

```
1)MOV EAX, 32
2)MOV EDX, 56
3)ADD EAX, EDX
4)MOV EBP, EAX
5)CMP EBP, 10
6)JGE 4470 <main+61>
7)MOV EAX, 0
8)CALL 4144 <printf@plt>
```

1&2)Partiamo dall'istruzione MOV (destinazione,sorgente), tale istruzione sposta il valore "sorgente" nella locazione "destinazione". Quindi per le prime due istruzioni sono equivalenti all'assegnazione di due interi (32 ,56) a i due registri (EAX , EDX):

EAX=32
EDX=56

3)La terza riga è un'istruzione di somma ADD che viene salvata nel registro EAX:

EAX=EAX+EDX=32+56=88
EDX=56

4)La quarta istruzione è di nuovo MOV ma differenza delle prime due, la sorgente in questo caso è una variabile che viene copiata e incollata nella destinazione.

EAX=88
EBP=EAX=88
EDX=56

5)La quinta riga esegue un'operazione di confronto [CMP destinazione,sorgente] nello specifico utilizza 2 variabili: ZF(zero flag) CF(carry flag) quindi non modifica nulla nei registri che inseriamo come campi. La tabella di verità di CMP ricorda molto quella dei resti interi

nelle divisioni distribuite (vedi slide). Nel dettaglio vediamo che la ZF risulta VERA se e solo se i valori sono uguali (x/x=1).
Altrimenti ZF è sempre FALSA.
CF invece è VERA se e solo se “sorgente”>”destinazione” ovvero se la divisione ha resto.
In questo caso è equivalente a dire:

CMP 88,10

Essendo 88>10 entrambe le variabili ZF e CF sono false.

EAX=88
EBP=88
EDX=56
ZF=0
CF=0

6)L’istruzione JGE è un'istruzione di salto che si concretizza se e solo se la destinazione è maggiore o uguale alla sorgente nella precedente istruzione CMP.
Come visto prima 88>=10 è VERO quindi viene attuato un salto di memoria all’indirizzo specificato: 0x1176 <main+61>

7)Tale istruzione serve a portare a zero il valore del registro EAX

EAX=0
EBP=88
EDX=56
ZF=0
CF=0

8)Infine CALL è l’istruzione utile a chiamare funzioni (in questo caso printf) che si concretizza prendendo in input il dato di memoria inserito 0x1030.

Se volessimo vedere tutto ciò come un codice C sarebbe abbastanza simile a:

```
a=32;
d=56;
a=a+d;
b=a;
…
if (b>10){
          }
printf(x);
```