Analisi codice

```
0x00001141 <+8>: mov EAX,0x20
```

1. viene caricato il valore decimale 32 nel registro EAX.

```
0x00001148 <+15>: mov EDX,0x38
```

2. viene caricato il valore decimale 58 nel registro EDX.

```
0x00001155 <+28>: add EAX,EDX
```

3. viene eseguita un'operazione di somma tra il valore in EAX (32) e il valore in EDX (58), e il risultato viene salvato nuovamente in EAX.

```
0x00001157 <+30>: mov EBP, EAX
```

4. il valore di EAX viene copiato nel registro EBP.

```
0x0000115a <+33>: cmp EBP,0xa
```

5. viene eseguita una comparazione tra il valore in EBP e il valore esadecimale a (decimale 10).

```
0x0000115e <+37>: jge 0x1176 <main+61>
```

6. viene eseguito un salto condizionale (jge) a 0x1176 <main+61> se il valore in EBP è maggiore o uguale a 10.

```
0x0000116a <+49>: mov eax,0x0
```

7. viene caricato il valore 0 nel registro EAX.

```
0x0000116f <+54>: call 0x1030 <printf@plt>
```

8. viene chiamata la funzione printf().

Codice in c

Questo è un potenziale codice in c che potrebbe dare come risultato quel codice in assembly:

```c
int main() {
  int a = 32;
  int b = 58;
  a += b;
  int c = a;
  if (c < 10) {
    printf("%d", c);
  }
  return 0;
}
```