# Pratica W23D1(2)

## Traccia:

EPICODE

**Esercizio**
Assembly - C

**Traccia:**

Dato il seguente codice assembly, provare a ricostruire le istruzioni originali in C

```
push   %ebp
mov    %esp,%ebp
sub    $0x8,%esp
call   80483e9 <bar>
leave
ret


push   %ebp
mov    %esp,%ebp
sub    $0x8,%esp
call   80483fb <baz>
call   8048400 <quux>
leave
ret
```

## Soluzione:

```
void foo() {
   int ebp_backup = ebp;  // Salva il valore corrente di ebp (base pointer)
   ebp = esp;          // Imposta ebp (base pointer) come il valore corrente di esp (stack pointer)
   esp -= 8;           // Sposta lo stack pointer di 8 byte verso il basso
   bar();              // Chiama la funzione bar()
   return;             // Ritorna al chiamante
}

void main() {
   int ebp_backup = ebp;  // Salva il valore corrente di ebp (base pointer)
```

```
    ebp = esp;          // Imposta ebp (base pointer) come il valore corrente di esp (stack pointer)
    esp -= 8;           // Sposta lo stack pointer di 8 byte verso il basso
    baz();              // Chiama la funzione baz()
    quux();             // Chiama la funzione quux()
    return;             // Ritorna al chiamante
}
```

**Spiegazione:**

1. **int ebp_backup** = ebp;: Questa istruzione salva il valore corrente di ebp (base pointer), che punta alla base del frame di stack corrente, in una variabile locale ebp_backup. Questo è comunemente usato per salvare lo stato di ebp prima di modificarlo.
2. **ebp = esp**;: Questa istruzione imposta ebp (base pointer) come il valore corrente di esp (stack pointer). In sostanza, questo aggiorna ebp per puntare al nuovo frame di stack corrente, di solito inizio di una nuova funzione.
3. **esp -= 8**;: Questa istruzione sposta lo stack pointer esp di 8 byte verso il basso. Questo può essere fatto per fare spazio per variabili locali o parametri di funzione.
4. **bar();**: Questa istruzione chiama la funzione bar().
5. **return**;: Questa istruzione ritorna al chiamante.