ESERCIZIO W23D1 PARTE 2

L’esercizio consiste nel covertire questo programma in assembly nel suo equivalente 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
```

//Primo set di istruzioni

void bar() {

// Corpo della funzione bar

}

int main() {

int ebp;

```
// push %ebp
// mov %esp, %ebp
int esp = ebp - 8;

bar();

// leave
// ret

return 0;
}


//Secondo set di istruzioni
void baz() {
// Corpo della funzione baz
}

void quux() {
// Corpo della funzione quux
}

int main() {
int ebp;
```

```
// push %ebp
// mov %esp, %ebp

int esp = ebp - 8;

baz();
quux();

// leave
// ret

return 0;
}
```

```
push   %ebp
mov    %esp,%ebp
pop    %ebp
ret


push   %ebp
mov    %esp,%ebp
mov    $0x0,%eax
movl   $0x1,(%eax)
pop    %ebp
ret


push   %ebp
mov    %esp,%ebp
and    $0xfffffff0,%esp
call   80483dc <foo>
mov    $0x0,%eax
leave
ret
```

//Primo set di istruzioni

{

// push %ebp

// C: Salva il valore corrente di ebp

int saved_ebp = ebp;

// mov %esp,%ebp

// C: Imposta ebp uguale a esp

ebp = esp;

```
// pop %ebp
// C: Ripristina il valore precedente di ebp
ebp = saved_ebp;

// ret
// C: Ritorna dal metodo
return;
}

// Secondo set di istruzioni
{
// push %ebp
int saved_ebp = ebp;

// mov %esp,%ebp
ebp = esp;

// mov $0x0,%eax
// C: Imposta eax a 0
eax = 0;

// movl $0x1,(%eax)
// C: Scrive il valore 1 all'indirizzo puntato da eax
// Nota: Questo è un comportamento indefinito in C perché eax è 0,
```

// quindi qui si sta tentando di scrivere all'indirizzo di memoria 0x0.

(int)eax = 1;

// pop %ebp

ebp = saved_ebp;

// ret

return;

}

// Terzo set di istruzioni

{

// push %ebp

int saved_ebp = ebp;

// mov %esp,%ebp

ebp = esp;

// and $0xfffffff0,%esp

// C: Allinea esp a un multiplo di 16

esp = esp & 0xFFFFFFF0;

// call 80483dc <foo>

// C: Chiama la funzione foo()

```
foo();

// mov $0x0,%eax
eax = 0;

// leave
// C: Ripristina esp e ebp ai loro valori precedenti
esp = ebp;
ebp = saved_ebp;

// Assembly: ret
return;
}
```