#include <stdio.h>

void main(void) {
    printf("No parameters\n");
}

mov esp, ebp
pop ebp

Leave will release frame:

RET pops return address

mov -16(%rbp), %rax
movq (%rax), %rax
movq %rax, %rsi
movl $.LC1, %edi
movl $0, %eax
call printf
leave

movq -16(%rbp), %rax
movq (%rax), %rax
movq %rax, %rsi
movl $.LC1, %edi
movl $0, %eax
call printf

mov -16(%rbp), %rax
movq (%rax), %rax
movq %rax, %rsi
movl $.LC1, %edi
movl $0, %eax
call printf
### PUSH -- Push Operand onto the Stack

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Instruction</th>
<th>Clocks</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF /6</td>
<td>pushw m16</td>
<td>5</td>
<td>Push memory word</td>
<td>pushw m16</td>
</tr>
<tr>
<td>FF /6</td>
<td>pushl m32</td>
<td>5</td>
<td>Push memory dword</td>
<td>pushl m32</td>
</tr>
<tr>
<td>50 +/r</td>
<td>pushw r16</td>
<td>2</td>
<td>Push register word</td>
<td>pushw %bx</td>
</tr>
<tr>
<td>50 +/r</td>
<td>pushl r32</td>
<td>2</td>
<td>Push register dword</td>
<td>pushl %ebx</td>
</tr>
<tr>
<td>68</td>
<td>pushw imm16</td>
<td>2</td>
<td>Push immediate word</td>
<td>pushw $0x7fff</td>
</tr>
<tr>
<td>68</td>
<td>pushl imm32</td>
<td>2</td>
<td>Push immediate dword</td>
<td>pushl $0x7fffffff</td>
</tr>
<tr>
<td>0E</td>
<td>pushw cs</td>
<td>2</td>
<td>Push CS</td>
<td>pushw %cs</td>
</tr>
<tr>
<td>16</td>
<td>pushw ss</td>
<td>2</td>
<td>Push SS</td>
<td>pushw %ss</td>
</tr>
<tr>
<td>1E</td>
<td>pushw ds</td>
<td>2</td>
<td>Push DS</td>
<td>pushw %ds</td>
</tr>
<tr>
<td>06</td>
<td>pushw es</td>
<td>2</td>
<td>Push ES</td>
<td>pushw %es</td>
</tr>
<tr>
<td>0F A0</td>
<td>pushw fs</td>
<td>2</td>
<td>Push FS</td>
<td></td>
</tr>
<tr>
<td>OF A8</td>
<td>pushw gs</td>
<td>2</td>
<td>Push GS</td>
<td></td>
</tr>
</tbody>
</table>

### MOVQ -- Move Quadword

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0F 6F /r</td>
<td>MOVQ mm, mm/m64</td>
<td>Move quadword from mm/m64 to mm.</td>
</tr>
<tr>
<td>0F 7F /r</td>
<td>MOVQ mm/m64, mm</td>
<td>Move quadword from mm to mm/m64.</td>
</tr>
<tr>
<td>F3 0F 7E</td>
<td>MOVQ xmm1, xmm2/m64</td>
<td>Move quadword from xmm2/mem64 to xmm1.</td>
</tr>
<tr>
<td>66 0F D6</td>
<td>MOVQ xmm2/m64, xmm1</td>
<td>Move quadword from xmm1 to xmm2/mem64.</td>
</tr>
</tbody>
</table>

**Description**

Copies a quadword from the source operand (second operand) to the destination operand (first operand). The source and destination operands can be MMX™ technology registers, XMM registers, or 64-bit memory locations. This instruction can be used to move a between two MMX registers or between an MMX register and a 64-bit memory location, or to move data between two XMM registers or between an XMM register and a 64-bit memory location. The instruction cannot be used to transfer data between memory locations.

When the source operand is an XMM register, the low quadword is moved; when the destination operand is an XMM register, the quadword is stored to the low quadword of the register, and the high quadword is cleared to all 0s.

**Example**

```assembly
movl $4, %eax
```

Move long the value 4 to eax register
```c
#include <stdio.h>
int fact(int n) {
    if(n<3) return n;
    return n*fact(n-1);
}

int main() {
    printf("fact=%d\n", fact(5));
    return 15;
}
```