IA32

32-bit registers:

|  |  |  |
| --- | --- | --- |
| EAX | Accumulator | General Purpose Registers |
| EBX |  |
| ECX |  |
| EDX |  |
| ESI |  | Memory Address Registers |
| EDI |  |
| EBP | Frame Pointer |
| ESP | Stack Pointer |
| EFLAGS | Flags |  |
| EIP | Instruction Pointer |

**Accessing IA32 Registers**:

|  |  |  |  |
| --- | --- | --- | --- |
| EAX (32) | | | |
|  |  | AX (16) | |
|  |  | AL (8) | AH (8) |

Instruction Examples: (L←R)

|  |
| --- |
| ADD EAX, EBX ; EAX = EAX + EBX  MOV EAX, EBX ; EAX = EBX  SHR EAX, 1 ; EAX >>= 1  MOV EAX, [EBX+4] ; EAX = Value in memory at address EBX+4  LEA EAX, [EBX\*ECX+8] ; Load effective address - useful for arithmetic |

Memory ← Memory & Memory ← Immediate are **not** allowed in IA32:

|  |
| --- |
| MOV [EAX], [EBX] ; Memory ← Memory **not** allowed.  MOV [EAX], 16 ; Memory ← Immediate **not** allowed. |

**Branching**:

CMP Compare

JMP Branch

JE Branch if equal

JNE Branch if not equal

**Signed**: **Unsigned**:

JL JB Branch if less than

JLE JBE Branch if less then or equal

JG JA Branch if greater than

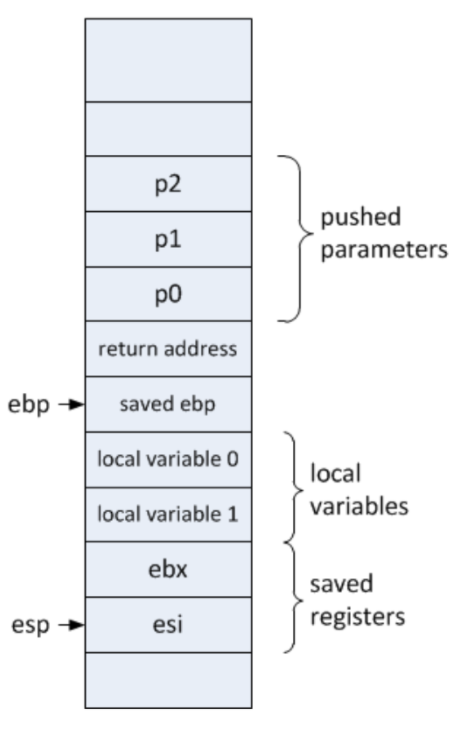
JGE JAE Branch if greater than or equal

Clear register:

|  |
| --- |
| XOR EAX, EAX |

Test if a register is zero:

|  |
| --- |
| TEST EAX, EAX  JE label |



### Calling Functions

1. Push parameters onto stack in **reverse** order (Pn-1, Pn-2, ... , P0).
2. CALL <function\_name>
3. PUSH EBP (Save previous frame pointer)
4. MOV EBP, ESP (Initialise a new frame pointer, i.e. Save stack w/o locals)
5. Decrement ESP to allocate local variables.
6. Save non-volatile registers being used on stack.
7. <FUNCTION BODY>
8. Restore non-volatile registers from stack.
9. MOV ESP, EBP (Remove locals from stack)
10. POP EPB (Restore previous frame pointer)
11. RET (Return from function)
12. Increment ESP to remove parameters.

Access relative to EBP (Frame Pointer).

P0 = [EBP+8], P1 = [EBP+12]

L0 = [EBP-4], L1 = [EBP-8]

### IA32 Calling Convention

|  |  |
| --- | --- |
| EAX | Function Result |
| EBX |  |
| ECX | Volatile |
| EDX |

It is the caller’s responsibility to remove parameters.

Parameters are pushed from right to left so that functions can handle arbitrary numbers of parameters. (Parameter 0 is always [EBP+8])

Example:

|  |
| --- |
| ; function(1, 2, 3);  PUSH 3  PUSH 2  PUSH 1  CALL function  ADD ESP, 12 ; Remove 3 parameters from stack. |

x64

64-bit registers:

|  |  |  |
| --- | --- | --- |
| RAX | Accumulator | General Purpose Registers |
| RBX |  |
| RCX |  |
| RDX |  |
| RSI |  | Memory Address Registers |
| RDI |  |
| RBP | Frame Pointer |
| RSP | Stack Pointer |
| R8 |  | General Purpose Registers |
| R9 |  |
| R10 |  |
| R11 |  |
| R12 |  |
| R13 |  |
| R14 |  |
| R15 |  |
| RFLAGS | Flags |  |
| RIP | Instruction Pointer |

**Accessing x64 Registers**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RAX (64) | | | | | | | |
|  |  |  |  | EAX (32) | | | |
|  |  |  |  |  |  | AX (16) | |
|  |  |  |  |  |  | AH (8) | AL (8) |

### x64 Calling Convention

|  |  |
| --- | --- |
| RAX | Function Result |
| RBX |  |
| RCX | Parameter 0 |
| RDX | Parameter 1 |
| RSI |  |
| RDI |  |
| RBP |  |
| RSP |  |
| R8 | Parameter 2 |
| R9 | Parameter 3 |
| R10 | Volatile |
| R11 |
| R12 |  |
| R13 |  |
| R14 |  |
| R15 |  |

Extra parameters are pushed onto stack (R → L).

Caller must allocate 32B (4 \* 8B) of shadow space on the stack before calling, and de-allocate after returning.