Simple Instructions

Simple Instructions

- **mov destination, source**
  - Moves data from one location to another
- We use Intel format throughout the book, with destination first
- Remember indirect addressing
  - [ebx] means the memory location pointed to by EBX

Table 5-4. `mov` Instruction Examples

Instruction	Description
`mov eax, ebx`	Copies the contents of EBX into the EAX register
`mov eax, 0x42`	Copies the value 0x42 into the EAX register
`mov eax, [0x4037C4]`	Copies the 4 bytes at the memory location 0x4037C4 into the EAX register
`mov eax, [ebx]`	Copies the 4 bytes at the memory location specified by the EBX register into the EAX register
`mov eax, [ebx+esi*4]`	Copies the 4 bytes at the memory location specified by the result of the equation `ebx+esi*4` into the EAX register

lea (Load Effective Address)

- lea destination, source
- lea eax, [ebx+8]
  - Puts ebx + 8 into eax
- Compare
  - mov eax, [ebx+8]
  - Moves the data at location ebx+8 into eax

Figure 5-5 shows values for registers EAX and EBX on the left and the information contained in memory on the right. EBX is set to 0xB30040. At address 0xB30048 is the value `0x20`. The instruction `mov eax, [ebx+8]` places the value `0x20` (obtained from memory) into EAX, and the instruction `lea eax, [ebx+8]` places the value `0xB30048` into EAX.

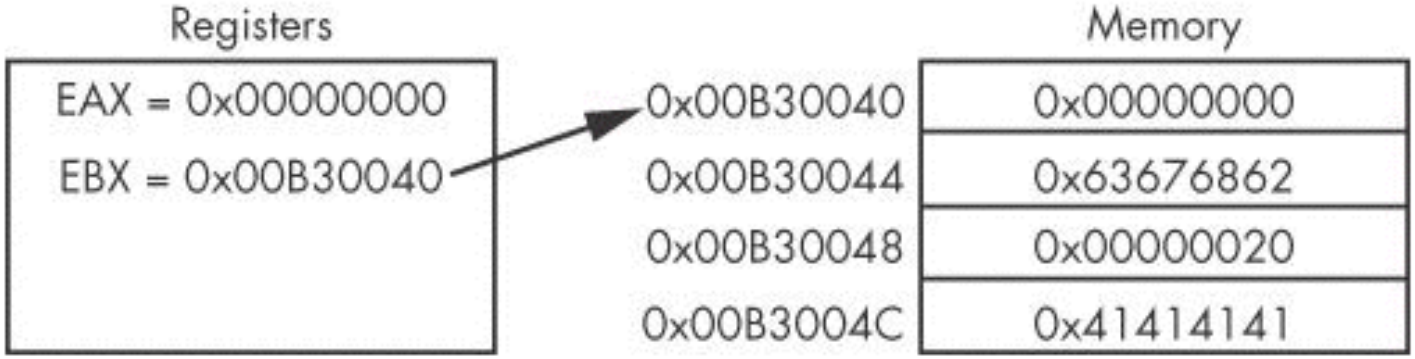

Figure 5-5. EBX register used to access memory

Arithmetic

- **sub** Subtracts
- **add** Adds
- **inc** Increments
- **dec** Decrements
- **mul** Multiplies
- **div** Divides

NOP

- Does nothing
- 0x90
- Commonly used as a **NOP Sled**
- Allows attackers to run code even if they are imprecise about jumping to it

The Stack

- Memory for functions, local variables, and flow control
- Last in, First out
- ESP (Extended Stack Pointer) – top of stack
- EBP (Extended Base Pointer) – bottom of stack
- PUSH puts data on the stack
- POP takes data off the stack

Other Stack Instructions

- To enter a function
  - Call or Enter
- To exit a function
  - Leave or Ret

Function Calls

- Small programs that do one thing and return, like printf()
- Prologue
  - Instructions at the start of a function that prepare stack and registers for the function to use
- Epilogue
  - Instructions at the end of a end of a function that restore the stack and registers to their state before the function was called

Stack Frames

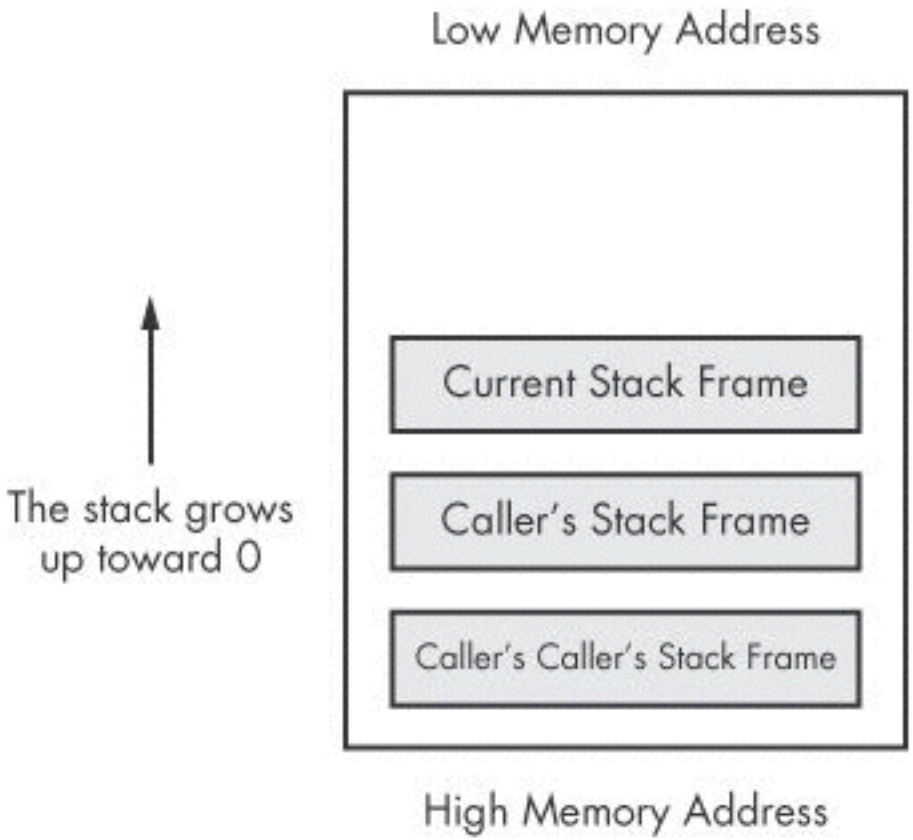

Figure 5-7. x86 stack layout

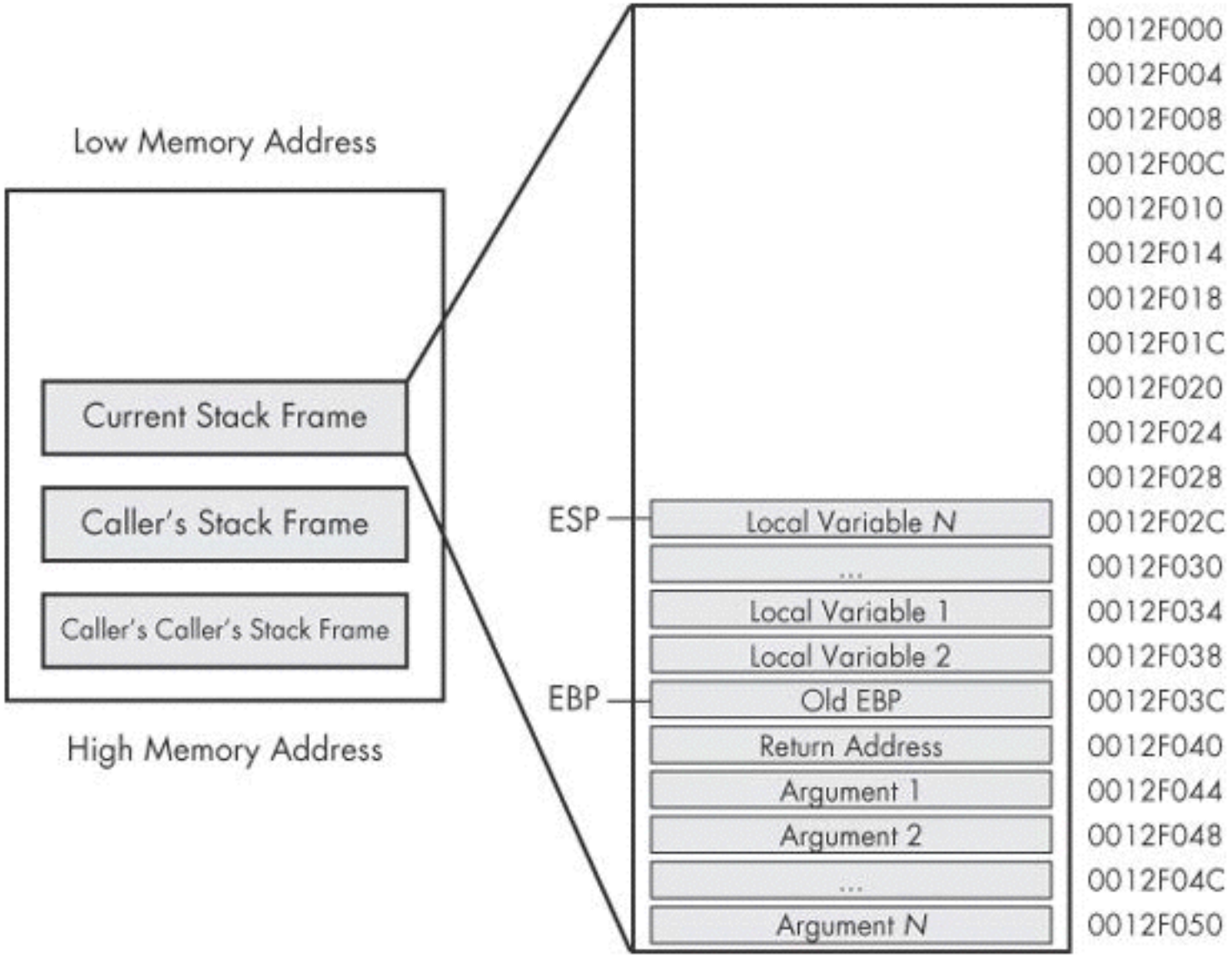

Figure 5-8. Individual stack frame

Conditionals

- test
  - Compares two values the way AND does, but does not alter them
  - test eax, eax
    - Sets Zero Flag if eax is zero
- cmp eax, ebx
  - Sets Zero Flag if the arguments are equal

Branching

- jz loc
  - Jump to loc if the Zero Flag is set
- jnz loc
  - Jump to loc if the Zero Flag is cleared

C Main Method

- Every C program has a main() function
- int main(int argc, char** argv)
  - argc contains the number of arguments on the command line
  - argv is a pointer to an array of names containing the arguments

Example

- cp foo bar
  - argc = 3
  - argv[0] = cp
  - argv[1] = foo
  - argv[2] = bar