x86 Assembly Code

Rupesh Nasre.

CS3300 Compiler Design
IIT Madras
August 2020

Based on the references one and two.

Assembly Code

- Just above the machine code
- In presence of assembler, the output of your compiler
- Useful for efficient, low-level programming
  - Reduces productivity
- We will use Linux-based assembler *as*.
  - Example of GAS. Also, Netwide assembler is popular (NASM).
  - On Windows, MASM is popular (uses Intel syntax).
  - Software interrupts and code libraries differ across OSs.

Intel Microprocessors

Microprocessor	Year of introduction	Number of transistors
4004	1971	2,250
8008	1972	3,500
8080	1974	6,000
8086	1978	29,000
80186	1982	55,000
80286	1982	134,000
80386	1985	275,000
80486	1989	1,180,235
Pentium	1993	3,100,000
Pentium II	1997	7,500,000
Pentium III	2000	21,000,000
Pentium 4	2000	42,000,000
Core	2006	184,000,000
Core 2	2006	291,000,000
i Series	2008	731,000,000
Xeon	Since 1998	8,000,000,000 (2017)

Registers

- 8 General-Purpose Registers (GPRs)
  - AX, BX, CX, DX, SI, DI, SP, BP
- 6 Segment Registers
  - SS, CS, DS, ES, FS, GS (not required for you)
- 1 Flags Register (EFLAGS: 32 bits)
  - Bit 0 is carry, 6 is zero, 7 is sign, 8 is trap (used in step-by-step debugging)
- 1 Instruction Pointer (EIP: 32 bits)
- x86-64 has additional registers.

Registers

Register	Accumulator			...
64-bit	RAX			...
32-bit		EAX		...
16-bit		AX		...
8-bit		AH	AL	...

Memory

- x86 architecture is little-endian.
- Least significant byte is written first in memory.

- 0x12345678 is written as

78	56	34	12	...

- Source: 3.c

Bye World! … exit

- **Source**: 2.s
- mov src, dst ; comment
  - Opcode and registers are case insensitive, labels are case sensitive.
  - Constants are prefixed with $.
  - Registers are prefixed with %.
  - Assembler directives start with a dot.
  - b=byte, w=word (16), l=long (int32), q=quad
  - int invokes an interrupt handler.
  - exit system call expects exit code in ebx register.

Hello World!

- **Source** 1.c
- .section .data starts data section.
- .section .text starts instructions section.
- .section .rodata is for read-only data.
- .globl marks a label used by the linker; so assembler should not tamper with it.
- _start / main are special labels, marking beginning of instructions when the program loads.

Max

- **Source**: 4.s
  - Assemble as: as 4.s -o 4.o
  - Link as: ld 4.o -o max
  - Run as: ./max; echo $?
  - Note that 0 marks the end of the number list.
  - Try changing 44 to 344.
  - Program is correct, but exit codes are 8-bit long (try return 300 in a .c program).

Addressing Modes

Addressing Mode	Example Instruction
Register	mov ax, bx
Immediate	mov $0xABCD, ax
Direct memory	.data my_var dw 0abcdh .code mov ax, [my_var]
Direct offset	byte_table db 12, 15, 16, 22 mov al, [byte_table + 2] mov al, byte_table[2]
Register indirect	mov ax, [di]

Instruction Set

Instruction Type	Example Instruction
Data transfer	movl $0x000F, %eax ; byte=8, word=16, long=32, quad=64

Control-flow	mov ecx, $5
	mov edx, $5
	cmp ecx, edx
	je equal
	;----------
	call fun
	;----------
	mov ecx, 5
	five:
	; the code here is executed 5 times
	loop five

Arithmetic	mov eax, [source] ; read low 32 bits
	mov edx, [source+4] ; read high 32 bits
	add [destination], eax ; add low 32 bits
	adc [destination+4], edx ; add high 32 bits, plus carry

Instruction Set

Instruction Type	Example Instruction
Data transfer	...
Control-flow	...
Arithmetic	...

Logic	movl $0x1, %edx ; edx ≔ 1
	movl $0x0, %ecx ; ecx ≔ 0
	andl %edx, %ecx ; ecx ≔ edx ∧ ecx
	; here ecx would be 0 because 1 ∧ 0 ⇔ 0

Shift and Rotate	movw $ff00,%ax # ax=1111.1111.0000.0000 shrw $3,%ax # ax=0001.1111.1110.0000 shlw $1,%ax # ax=0011.1111.1100.0000
Others	push, pop, pusha, popa, pushf, popf, stc, clc, rdtsc
Interrupts	int 0x0a

Example on addressing modes

- movl -8(%ebp, %edx, 4), %eax # Full ex.: eax = *(ebp + (edx * 4) - 8)
- movl -4(%ebp), %eax # Typical ex.: eax = a stack variable
- movl (%ecx), %edx # Register indirect to register
- leal 8(,%eax,4), %eax # Arithmetic: eax = 4 * eax + 8
- leal (%edx,%eax,2), %eax # Arithmetic: eax = 2 * eax + edx

Notes

- *as* assembler allows C/C++ style comments.
  - /* … */ and //.
  - It also allows comments using #.
- SSE/AVX extensions allow 128, 256, 512 registers (e.g., XMM, YMM and ZMM).
- 8086 had 20-bit address. But registers were 16 bits. Hence, a combination of registers was used (Segment:Offset or CS:IP). With the current Flat Addressing (32 bits), this is not necessary.