X86 64 Assembly

Introduction

Kumaravel Jagasivamani
Mar. 6, 2024

Overview

- Registers
- ALU
- Instruction Memory
- Data Memory
- Simplified Processor Design
- Instruction Set Reference
- Instruction Format
- Opcode
- Instruction Details
- Running Assembler
- Stack
- Calling Functions

Registers

- 16 registers to store/load values:
  - RAX, RBX, RCX, RDX
  - RSI, RDI, RSP, RBP
  - R8~R15
- Some registers are used for other purposes as well
  - RAX: function return value
  - RSP: stack pointer

ALU

- Arithmetic operations
  - Add, Divide, Multiply, Compare, …
- Logical operations
  - XOR, AND, Shift, …
- Transfer operations
  - Move, …
- Flow control operations
  - Jump, Call, …
- Updates RFLAGS register
  - ZF: Zero Flag
  - SF: Sign Flag
  - CF (Unsigned) / OF (Signed): Overflow detection

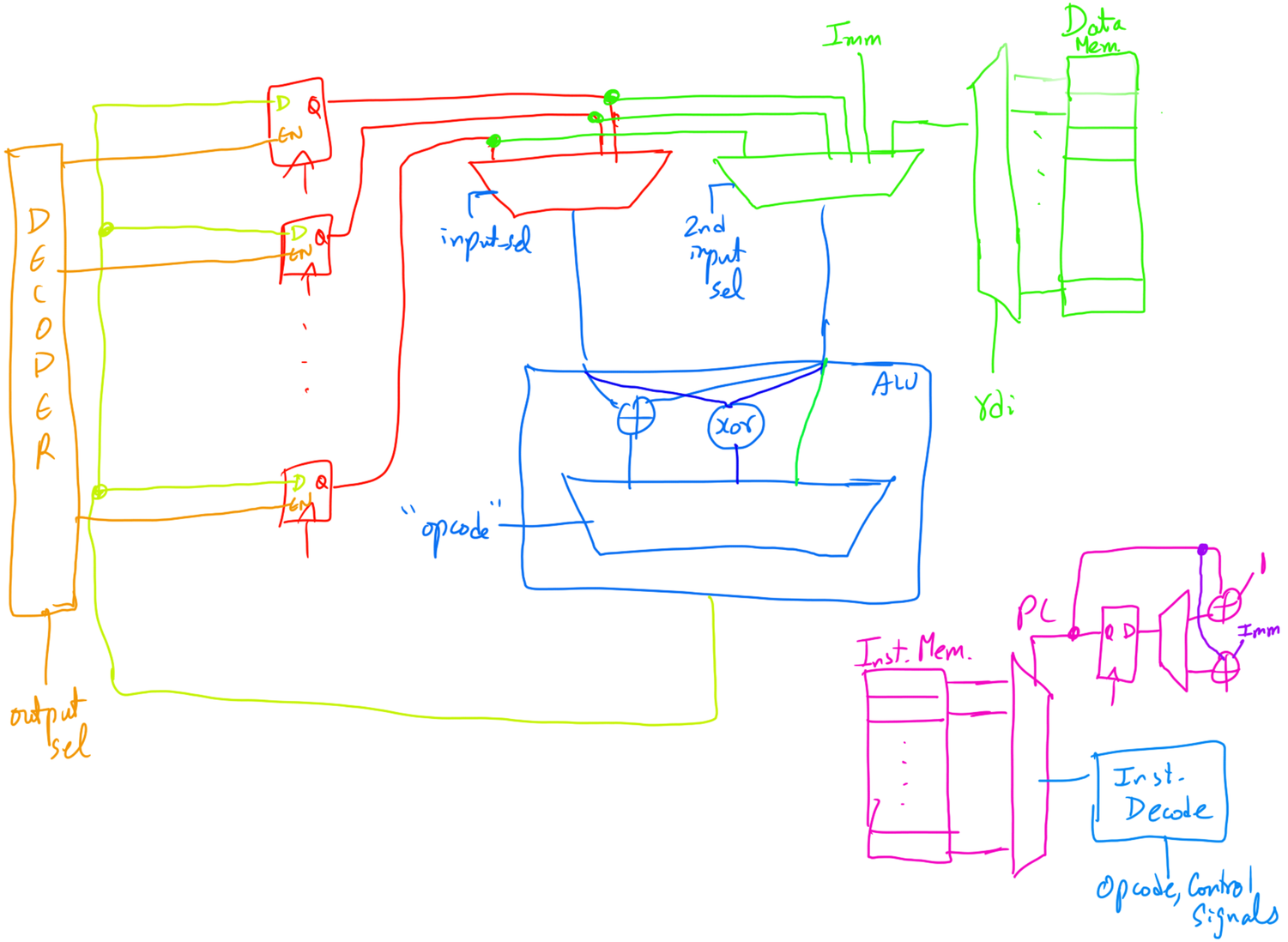
Instruction Memory

- Stores all the instructions to be executed
- Pointer to current instruction:
  - Program Counter: RIP register

Data Memory

- Allocate space in memory and assign label
  - input_val: db 50
- Pre-load memory location with constant
  - Warning if memory space isn’t initialized
- Load register value with label
  - Pointer to memory location
  - mov rdi, input_val
- Access memory: [rdi]

Simplified Processor Design



X86 64 Instruction Set Reference

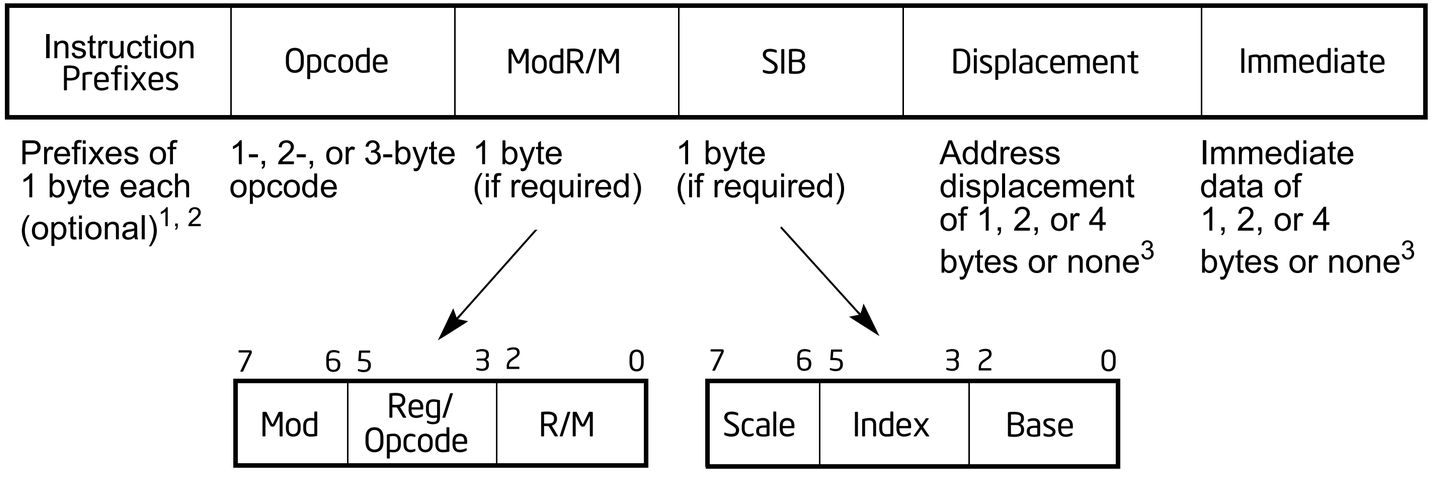
Volume 2

- https://cdrdv2.intel.com/v1/dl/getContent/671110

Instruction Format

Section 2.1

- Variable size of instruction:
  - 1~15 bytes
-



1. The REX prefix is optional, but if used must be immediately before the opcode; see Section 2.2.1, “REX Prefixes” for additional information.
2. For VEX encoding information, see Section 2.3, “Intel® Advanced Vector Extensions (Intel® AVX)”.
3. Some rare instructions can take an 8B immediate or 8B displacement.

Figure 2-1. Intel 64 and IA-32 Architectures Instruction Format

Opcodes
Appendix A.3

- 100s of instructions supported

Table A-2. One-byte Opcode Map: (00H — F7H) *

	0	1	2	3	4	5	6	7
0	ADD						PUSH ES^{i64}	POP ES^{i64}
	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz		
1	ADC						PUSH SS^{i64}	POP SS^{i64}
	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz		
2	AND						SEG=ES (Prefix)	DAA^{i64}
	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz		
3	XOR						SEG=SS (Prefix)	AAA^{i64}
	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz		
4	INC^{i64} general register / REX^{o64} Prefixes							
	eAX REX	eCX REX.B	eDX REX.X	eBX REX.XB	eSP REX.R	eBP REX.RB	eSI REX.RX	eDI REX.RXB
5	PUSH^{d64} general register							
	rAX/r8	rCX/r9	rDX/r10	rBX/r11	rSP/r12	rBP/r13	rSI/r14	rDI/r15
6	PUSHA^{i64}/	POPA^{i64}/	BOUND^{i64}	ARPL^{i64}	SEG=FS	SEG=GS	Operand	Address

Instruction Details

Chapters 3-6

- Add instruction

ADD—Add

Opcode	Instruction	Op/ En	64-bit Mode	Compat/ Leg Mode	Description
04 ib	ADD AL, imm8	I	Valid	Valid	Add imm8 to AL.
05 iw	ADD AX, imm16	I	Valid	Valid	Add imm16 to AX.
05 id	ADD EAX, imm32	I	Valid	Valid	Add imm32 to EAX.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3
RM	ModRM:reg (r, w)	ModRM:r/m (r)	N/A
MR	ModRM:r/m (r, w)	ModRM:reg (r)	N/A
MI	ModRM:r/m (r, w)	imm8/16/32	N/A
I	AL/AX/EAX/RAX	imm8/16/32	N/A

Description

Adds the destination operand (first operand) and the source operand (second operand) and then stores the result in the destination operand. The destination operand can be a register or a memory location; the source operand can be an immediate, a register, or a memory location. (However, two memory operands cannot be used in one

Operation

DEST := DEST + SRC;

Flags Affected

The OF, SF, ZF, AF, CF, and PF flags are set according to the result.

Running Assembler

- Log-in to UMBC server:
  - ssh gl.umbc.edu
- Compiling:
  - nasm -f elf64 <filename>.asm
  - gcc -m64 -o <filename> <filename>.o
- Running:
  - ./<filename>

Stack

- Store temporary information in separate memory region
- No need to pre-allocate space for data
- Push: Add value on top of stack
- Pop: Read value on top of stack and discard it
- Can only access the top-most value in stack

Calling functions

- Inputs:
  - RDI
  - RSI
  - …
- Outputs:
  - RAX
  - …
- Responsibility of saving registers (in stack):
  - Callee: RSP, RBX, RBP, R12~R15
  - Caller: RAX, RDI, RSI, RDX, RCX, R8~R11