RUTGERS

CS211 Computer Architecture

Fall 2020

Recitation 7

Today

- Intro to Assembly

Assembly Overview

- Assembly code is readable machine code – what the processor executes
- Consists of
  - Registers
  - Instructions
  - Addressing modes

Registers

- Eliminates need to access memory by acting as storage
- Used for storing different types of data and carry out instructions
- Different types
  - Ex: General purpose registers
    - Data registers
  - Pointer registers
    - Stack pointer
    - Base pointer

Instructions

- In assembly languages, syntax matters
  - Do not confuse with other x86 syntax when you are using Google

    <instruction> <source> <destination>

  - Known as AT&T assembly – do not get confused when Googling

Ex: `mov %eax, %ebx` □ moves contents of %eax into %ebx

Instructions

- Can’t move directly from memory to memory
  - Can move
    - Register □ address
    - Address □ register
    - Register □ register

NOT address □ address (must use a register)

Examples of MOV

- mov %ecx, %eax eax now has the data in ecx
- mov 8(%ecx), %eax eax now stores what ever is in the ADDRESS at **%ecx + 8**
- mov 8(%ecx, %ebx), %eax eax now has whatever is in the ADDRESS at **%ecx+%ebx+8**
- mov 8(%ecx, %ebx, 2), %eax eax now has whatever is in ADDRESS at **%exc + %ebx * 2 + 8**

Addressing Modes

Immediate

- Operand is a literal value
- mov $4, eax

Direct

- Operand is literal address
- mov %eax, 0x65 □ move contents of eax into address 0x65

Indirect

- Refer to address held in register
- mov (%ebx), %eax □ move whatever is at the address stored in ebx to eax

Register

- Contents of on register to the other
- mov %ebx, %eax □ if ebx = 5, move 5 into eax

Index

- Add/sub address values
- mov (%eax, %ebx), %eax □ move whatever is at address of eax+ebx into eax

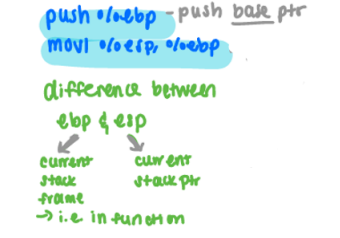
** how instructions access memory (as mentioned in previous slides)

Quick Note about Pointers

- Stack pointer
  - esp,rsp
  - Can change throughout frame
  - Will see

    ```
    sub <value>, esp
    ```

    So that esp now points to next byte after top of stack
- Base pointer
  - ebp, rbp
  - Base pointer for currents stack frame
  - Will not change throughout frame
  - Can access parameters as offset of ebp



From Lecture

%edx	0xf000
%ecx	0x100

Expression	Computation	Address
0x8(%edx)	0xf000 + 0x8	0xf008
(%edx,%ecx)	0xf000 + 0x100	0xf100
(%edx,%ecx,4)	0xf000 + 4*0x100	0xf400
0x80(,%edx,2)	2*0xf000 + 0x80	0x1e080

Quick C to Assembly Demo

- gdb disassemble
- Converting to .s file
  - gcc <c file> -S