CS 33: Introduction to Computer Organization

TA: Aalisha Dalal
LA: Jonathan Myong
Office Hours: Friday, 9:30-11:30AM

Focus of the Discussion

- Machine Level Basics
- Machine Level Control Instructions
- Worksheet Problems

Big Endian vs. Little Endian

Big Endian	**Little Endian**
- Numbers are stored in order in which they are printed	- Numbers are stored in reverse order
- Sign bit can be efficiently observed	- Lowest byte offset for 1, 2 or 4 byte number remains the same. - 1:1 relation b/w offset and byte number

Why Assembly Code?

- It is the closest to being a readable code which gives keen insights into the exact nature of machine code.
- Deepen your understanding of how the computer works!

How does Assembly code appear?

```
Dump of assembler code for function func4:
=> 0x0000000000400fe2 <+0>:     sub    $0x8,%rsp
   0x0000000000400fe6 <+4>:     mov    %edx,%eax
   0x0000000000400fe8 <+6>:     sub    %esi,%eax
   0x0000000000400fea <+8>:     mov    %eax,%ecx
   0x0000000000400fec <+10>:    shr    $0x1f,%ecx
   0x0000000000400fef <+13>:    add    %ecx,%eax
   0x0000000000400ff1 <+15>:    sar    %eax
   0x0000000000400ff3 <+17>:    lea    (%rax,%rsi,1),%ecx
   0x0000000000400ff6 <+20>:    cmp    %edi,%ecx
   0x0000000000400ff8 <+22>:    jle    0x401006 <func4+36>
   0x0000000000400ffa <+24>:    lea    -0x1(%rcx),%edx
   0x0000000000400ffd <+27>:    callq  0x400fe2 <func4>
   0x0000000000401002 <+32>:    add    %eax,%eax
   0x0000000000401004 <+34>:    jmp    0x40101b <func4+57>
   0x0000000000401006 <+36>:    mov    $0x0,%eax
   0x000000000040100b <+41>:    cmp    %edi,%ecx
   0x000000000040100d <+43>:    jge    0x40101b <func4+57>
   0x000000000040100f <+45>:    lea    0x1(%rcx),%esi
   0x0000000000401012 <+48>:    callq  0x400fe2 <func4>
   0x0000000000401017 <+53>:    lea    0x1(%rax,%rax,1),%eax
   0x000000000040101b <+57>:    add    $0x8,%rsp
   0x000000000040101f <+61>:    retq
```

Swap - Example

```
void swap
    (long *xp, long *yp)
{
   long t0 = *xp;
   long t1 = *yp;
   *xp = t1;
   *yp = t0;
}
```

```
swap:
    movq    (%rdi), %rax  # t0 = *xp
    movq    (%rsi), %rdx  # t1 = *yp
    movq    %rdx, (%rdi)  # *xp = t1
    movq    %rax, (%rsi)  # *yp = t0
    ret
```

Demos - Assembly code

Link to the Worksheet Problems

https://tinyurl.com/y4mqb7bx

Appendix - I

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

Appendix - II


Source
Dest
Src,Dest
C Analog
movq
Imm
Reg
movq $0x4,%rax
temp = 0x4;
Mem
movq $-147,(%rax)
*p = -147;
Reg
Reg
movq %rax,%rdx
temp2 = temp1;
Mem
movq %rax,(%rdx)
*p = temp;
Mem
Reg
movq (%rax),%rdx
temp = *p;

Questions