```
Handout for CS 372H
Lecture 2
21 January 2010

Some stuff to accompany lecture: pseudocode, memory layout, gcc calling
convention example.....

[Credit to Frans Kaashoek, Robert Morris, and Nickolai Zeldovich.]

1. using the IN and OUT instructions

    writing a byte to the parallel port (e.g., a line printer):

    #define DATA_PORT    0x378
    #define STATUS_PORT  0x379
    #define BUSY 0x80
    #define CONTROL_PORT 0x37A
    #define STROBE 0x01

    void
    lpt_putc(int c)
    {
      /* wait for printer to consume previous byte */
      while((inb(STATUS_PORT) & BUSY) == 0)
        ;

      /* put the byte on the parallel lines */
      outb(DATA_PORT, c);

      /* tell the printer to look at the data */
      outb(CONTROL_PORT, STROBE);
      outb(CONTROL_PORT, 0);
    }
```

```
2. The PC’s physical memory map:

        +------------------+  <- 0xFFFFFFFF (4GB)
        |      32-bit      |
        |  memory mapped   |
        |     devices      |
        |                  |
        /\/\/\/\/\/\/\/\/\/\

        /\/\/\/\/\/\/\/\/\/\
        |                  |
        |      Unused      |
        |                  |
        +------------------+  <- depends on amount of RAM
        |                  |
        |                  |
        | Extended Memory  |
        |                  |
        |                  |
        +------------------+  <- 0x00100000 (1MB)
        |     BIOS ROM     |
        +------------------+  <- 0x000F0000 (960KB)
        |  16-bit devices, |
        |  expansion ROMs  |
        +------------------+  <- 0x000C0000 (768KB)
        |   VGA Display    |
        +------------------+  <- 0x000A0000 (640KB)
        |                  |
        |    Low Memory    |
        |                  |
        +------------------+  <- 0x00000000
```

```
3. Example

    Here is the C code:
        int main(void) { return f(8)+1; }
        int f(int x) { return g(x); }
        int g(int x) { return x+3; }

    The assembly code:

        _main:
                        prologue

            pushl %ebp
            movl %esp, %ebp

                        body
            pushl $8
            call _f
            addl $1, %eax

                        epilogue
            movl %ebp, %esp
            popl %ebp
            ret

        _f:
                        prologue

            pushl %ebp
            movl %esp, %ebp

                        body
            pushl 8(%esp)
            call _g

                        epilogue
            movl %ebp, %esp
            popl %ebp
            ret

    <small version of _g>:
            movl 4(%esp), %eax
            addl $3, %eax
            ret

    <longer version of _g>:
                        prologue
            pushl %ebp
            movl %esp, %ebp

                        save %ebx
            pushl %ebx

                        body
            movl 8(%ebp), %ebx
            addl $3, %ebx
            movl %ebx, %eax

                        restore %ebx
            popl %ebx

                        epilogue
            movl %ebp, %esp
            popl %ebp
            ret
```

```
4. pseudocode for Bochs CPU simulation

    for (;;) {
        read_instruction();
        switch (decode_instruction_opcode()) {
        case OPCODE_ADD:
            int src = decode_src_reg();
            int dst = decode_dst_reg();
            regs[dst] = regs[dst] + regs[src];
            break;
        case OPCODE_SUB:
            int src = decode_src_reg();
            int dst = decode_dst_reg();
            regs[dst] = regs[dst] − regs[src];
            break;
        ...
        }
        eip += instruction_length;
    }

5. simulate PC’s physical memory map

    #define KB      1024
    #define MB      1024*1024

    #define LOW_MEMORY  640*KB
    #define EXT_MEMORY  10*MB

    uint8_t low_mem[LOW_MEMORY];
    uint8_t ext_mem[EXT_MEMORY];
    uint8_t bios_rom[64*KB];

    uint8_t read_byte(uint32_t phys_addr) {
        if (phys_addr < LOW_MEMORY)
            return low_mem[phys_addr];
        else if (phys_addr >= 960*KB && phys_addr < 1*MB)
            return bios_rom[phys_addr − 960*KB];
        else if (phys_addr >= 1*MB && phys_addr < 1*MB+EXT_MEMORY) {
            return ext_mem[phys_addr−1*MB];
        else ...
    }

    void write_byte(uint32_t phys_addr, uint8_t val) {
        if (phys_addr < LOW_MEMORY)
            low_mem[phys_addr] = val;
        else if (phys_addr >= 960*KB && phys_addr < 1*MB)
            ; /* ignore attempted write to ROM! */
        else if (phys_addr >= 1*MB && phys_addr < 1*MB+EXT_MEMORY) {
            ext_mem[phys_addr−1*MB] = val;
        else ...
    }
```