

```

1 Handout for CS 372H
2 Lecture 2
3 20 January 2011
4
5 Some stuff to accompany lecture: pseudocode, memory layout, gcc calling
6 convention example....
7
8 [Credit to Frans Kaashoek, Robert Morris, and Nickolai Zeldovich.]
9
10 1. using the IN and OUT instructions
11
12     writing a byte to the parallel port (e.g., a line printer):
13
14     #define DATA_PORT    0x378
15     #define STATUS_PORT  0x379
16     #define BUSY 0x80
17     #define CONTROL_PORT 0x37A
18     #define STROBE 0x01
19
20     void
21     lpt_putc(int c)
22     {
23         /* wait for printer to consume previous byte */
24         while((inb(STATUS_PORT) & BUSY) == 0)
25             ;
26
27         /* put the byte on the parallel lines */
28         outb(DATA_PORT, c);
29
30         /* tell the printer to look at the data */
31         outb(CONTROL_PORT, STROBE);
32         outb(CONTROL_PORT, 0);
33     }
34

```

```

35 2. The PC's physical memory map:
36
37     +-----+
38     | 32-bit | <- 0xFFFFFFFF (4GB)
39     | memory |
40     | mapped |
41     | devices|
42     +-----+
43     ^^^^^^
44     ^^^^^^
45     | Unused |
46     +-----+ <- depends on amount of RAM
47
48     | Extended Memory |
49
50
51
52
53
54     +-----+ <- 0x00100000 (1MB)
55     | BIOS ROM |
56     +-----+ <- 0x000F0000 (960KB)
57     | 16-bit devices, |
58     | expansion ROMs |
59     +-----+ <- 0x000C0000 (768KB)
60     | VGA Display |
61     +-----+ <- 0x000A0000 (640KB)
62
63     | Low Memory |
64     +-----+ <- 0x00000000
65
66

```

Jan 20, 11 16:32

I02-handout.txt

Page 3/4

```

67 3. Example
68
69 Here is the C code:
70 int main(void) { return f(8)+1; }
71 int f(int x) { return g(x); }
72 int g(int x) { return x+3; }
73
74 The assembly code:
75
76 _main:
77     prologue
78
79     pushl %ebp
80     movl %esp, %ebp
81
82     body
83     pushl $8
84     call _f
85     addl $1, %eax
86
87     epilogue
88     movl %ebp, %esp
89     popl %ebp
90     ret
91
92 _f:
93     prologue
94
95     pushl %ebp
96     movl %esp, %ebp
97
98     body
99     pushl 8(%esp)
100    call _g
101
102    epilogue
103    movl %ebp, %esp
104    popl %ebp
105    ret
106
107 <small version of _g>:
108 movl 4(%esp), %eax
109 addl $3, %eax
110 ret
111
112 <longer version of _g>:
113     prologue
114     pushl %ebp
115     movl %esp, %ebp
116
117     save %ebx
118     pushl %ebx
119
120     body
121     movl 8(%ebp), %ebx
122     addl $3, %ebx
123     movl %ebx, %eax
124
125     restore %ebx
126     popl %ebx
127
128     epilogue
129     movl %ebp, %esp
130     popl %ebp
131     ret
132

```

Jan 20, 11 16:32

I02-handout.txt

Page 4/4

```

133 4. Emulation of CPU in software
134
135 for (;;) {
136     read_instruction();
137     switch (decode_instruction_opcode()) {
138     case OPCODE_ADD:
139         int src = decode_src_reg();
140         int dst = decode_dst_reg();
141         regs[dst] = regs[dst] + regs[src];
142         break;
143     case OPCODE_SUB:
144         int src = decode_src_reg();
145         int dst = decode_dst_reg();
146         regs[dst] = regs[dst] - regs[src];
147         break;
148     ...
149     }
150     eip += instruction_length;
151 }
152
153
154
155
156 5. Emulate PC's physical memory map
157
158 #define KB      1024
159 #define MB      1024*1024
160
161 #define LOW_MEMORY  640*KB
162 #define EXT_MEMORY  10*MB
163
164 uint8_t low_mem[LOW_MEMORY];
165 uint8_t ext_mem[EXT_MEMORY];
166 uint8_t bios_rom[64*KB];
167
168 uint8_t read_byte(uint32_t phys_addr) {
169     if (phys_addr < LOW_MEMORY)
170         return low_mem[phys_addr];
171     else if (phys_addr >= 960*KB && phys_addr < 1*MB)
172         return bios_rom[phys_addr - 960*KB];
173     else if (phys_addr >= 1*MB && phys_addr < 1*MB+EXT_MEMORY) {
174         return ext_mem[phys_addr-1*MB];
175     } else ...
176 }
177
178 void write_byte(uint32_t phys_addr, uint8_t val) {
179     if (phys_addr < LOW_MEMORY)
180         low_mem[phys_addr] = val;
181     else if (phys_addr >= 960*KB && phys_addr < 1*MB)
182         /* ignore attempted write to ROM! */
183     else if (phys_addr >= 1*MB && phys_addr < 1*MB+EXT_MEMORY) {
184         ext_mem[phys_addr-1*MB] = val;
185     } else ...
186 }
187
188

```