Introduction to Computer Systems
2012 Fall Midterm Examination

Name_____ Student No._____ Score_____

Problem 1: (12 points)

1.

2.

3.

Problem 2: (16 points)

1.

2.

3.

4.

Problem 3: (27 points)

1 [1]
[2]
[3]
[4]
[5]
[6]

2 int srcA = [

];

int srcB = [

];

int dstM = [

];

3.

Problem 4: (20 points)

[1]	[2]
[3]	[4]
[5]	[6]
[7]	[8]
[9]	[10]

Problem 5: (12 points)

Problem 6: (13 points)

Problem 1: (12 points)

```
1  #include “csapp.h”
2
3  void Print(char c) { putchar(c); fflush(stdout); }
4
5  int main(void)
6  {
7      pid_t pid;
8      char c = ‘A’;
9
10     if ((pid = Fork()) == 0) {
11         Print(++c);
12         return 0;
13     }
14     Print(c);
15     waitpid(pid, NULL, 0);
16     Print(‘;’);
17
18     if (Fork() && Fork())
19         while (waitpid(-1, NULL, 0) > 0) Print(‘X’);
20
21     return 0;
22 }
```

1. After executing the statement in **line 16**, list all possible outputs before the semicolon (4’).

2. How many processes are created by the statement in **line 18**? Why? (4’).

3. How many ‘X’ are printed by the statement in **line 19**? Why? (4’)

Problem 2: (16points)

Consider the following code.

```
1  #include “csapp.h”
2
3  int n = 1; int a[] = { 12, -7, 99, 16 };
4  void handler_usr1(int sig) { n -= 3; }
5  void handler_usr2(int sig) { n += 4; }
6  void handler_chld(int sig) { printf(“p:%d\n”, a[n]); }
7  void handler_int(int sig) { printf(“c:%d\n”, a[n]); exit(0); }
8
9  int main(void) {
10     pid_t pid;
11
12     signal(SIGUSR1, handler_usr1);
13     signal(SIGUSR2, handler_usr2);
14     signal(SIGINT, handler_int);
15     signal(SIGCHLD, handler_chld);
16
17     if ((pid = Fork()) == 0) {
18        n++;
19        while (1) pause();
20     } else {
21        kill(pid, SIGUSR1);
22        kill(pid, SIGUSR2);
23        kill(pid, SIGINT);
24        while (1) pause();
25     }
26
27     return 0;
28 }
```

1. When ‘**handler_chld**’ is called in the parent process, what are the values of ‘**n**’ in the **two** processes (parent/child) respectively? (2’*2=4’)

   [Kernel calls signal handlers according to the order of kills]

2. Suppose ‘**printf’** would immediately output the number due to the presence of ‘**\n**’. What is the output of this program? (**NOTE**: Pay attention to the order of the lines) (4’)

3. Suppose the **PID** of the parent process is **1000**, design a series of ‘**kill**’ command (in shell) to make the program output ‘**c:12**’. (4’)

4. If you carelessly omit the **line 22**, what fault may occur in the program? Work out a way to handle this fault gracefully. (4’)

Problem 3: (27points)

Suppose we add a new instruction **XLAT** to Y86 instruction sets, it uses the following encoding.

Byte	0	
XLAT	C	0

The function of **XLAT** instruction is to change the **%eax** register from the table index to the table entry **without updating condition codes** (CC). In other words, it moves the 32-bit word to **%eax** addressed by **%ebx + %eax**. **%eax** should be the unsigned index into a table addressed by **%ebx**.

1. Fill in the function of each stage for **XLAT** instruction in Y86 sequential implementation. (2' * 6 = 12')

Stage	XLAT
Fetch	icode:ifun ← M_1[PC] [1]
Decode	[2] [3]
Execute	[4]
Memory	[5]
Write Back	[6]
PC Update	PC ← valP

2. The original SEQ implementation of Y86 should be modified to support the **XLAT** instruction, including **srcA**, **srcB**, **dstM** and so on. Please redesign the following circuit logics using HCL according to your answer to Part A. (The original version is provided below, and you can use "**IXLAT**" to denote **XLAT** instruction and "**REAX**", "**REBX**" to denote **%eax** and **%ebx** respectively) (3' * 3 = 9')

Original version:

```
int srcA = [
   icode in { IRRMOVL, IRMMOVL, IOPL, IPUSHL } : rA;
   icode in { IPOPL, IRET } : RESP;
   1 : RNONE;
];
int srcB = [
   icode in { IOPL, IRMMOVL, IMRMOVL } : rB;
   icode in { IPUSHL, IPOPL, ICALL, IRET } : RESP;
   1 : RNONE;
];
int dstM = [
   icode in { IMRMOVL, IPOPL } : rA;
   1 : RNONE;
];
```

3. Besides **srcA**, **srcB** and **dstM**, what other circuit logics should be modified to support **XLAT** instruction? Please select them from the following list. (6')

Name	**Description**
need_regids	Does fetched instruction require a regid byte?
need_valC	Does fetched instruction require a constant word?
instr_valid	Is fetched instruction valid?
srcA	What register should be used as the A source?
srcB	What register should be used as the B source?
dstE	What register should be used as the E destination?
dstM	What register should be used as the M destination?
aluA	Select input A to ALU
aluB	Select input B to ALU
alufun	Set the ALU function
set_cc	Should the condition codes be updated?
mem_read	Set read control signal
mem_write	Set write control signal
mem_addr	Select memory address
mem_data	Select memory input (write) data
new_pc	What address should instruction be fetched at?

Problem 4: (20points)

Suppose we have the following assembly code. This program will be executed on a five stage pipeline processor, which has forwarding technology and control mechanism (such as **stall** and **bubble**).

```
0x000   irmovl  $4, %eax
0x006   addl  %eax, %ebx
0x008   call  test
0x00d   mrmovl  0x4(%ebx), %ecx
0x013   addl  %eax, %ecx
0x020 .pos  0x20
0x020 test:
0x020   ret
0x021   xor  %eax, %eax
```

Initially, the value of **%esp** is **0x020**, and all the other registers have value **$0**. And, some values of areas of memory are:

address	value
0x0	0x1
0x4	0x2
0x8	03

Assume in cycle 0, no instruction is executed, and in cycle 1, the first instruction is fetched.

	1	2	3	4	5	6	7
irmovl $4, %eax	F	D	E	M	W		
addl %eax, %ebx		F	D	E	M	W	
...	...	...	...	...	...	...	...

Please fill in the blanks accoding to assembly code. You need write down the values that different stage has in different cycle (**NOTE**: do not just fill in the register name). (2' * 10 = 20')

	cycle 3	cycle 5	cycle 7	cycle 8	cycle 10	cycle 11
F	f_pc=0x8	f_pc=[1]	f_pc=[3]	f_pc=[5]		
D	d_valA=4 d_valB=0				d_valA=[6] d_valB=[7]	d_valA=[9] d_valB=[10]
E		e_valE=[2]			e_valE=[8]	
M			m_valM=[4]			
W						

Problem 5: (12points)

Suppose we have a block of code, but the performance is not very good. Please try to optimize the function **foo** using the technologies learnt in the ICS class.

```
typedef int data_t;

typedef struct {
   int len;
   data_t *data;
}*vec_ptr;

void foo(vec_ptr v, data_t *dest)
{
   for(int i = 1; i < get_vec_length(v); i++)
      *dest *= v->data[i] * 2;
   for(int i = 1; i < get_vec_length(v); i++)
      swap(&v->data[i - 1], &v->data[i])
}

/* DON'T modify the following functions */
void swap(data_t *px, data_t *py){
   data_t temp;
   temp = *px;  *px = *py;  *py = temp;
}

int get_vec_length(vec_ptr v)
{
   return v->len;
}
```

Problem 6: (13points)

```
.Loop
  movss (%rax, %rdx, 4), %xmm0
  mulss %rsi, %xmm1
  addq %xmm0, %xmm1
  addq $-1, %rdx
  cmpq %rcx, %rdx
  jg .Loop
```

Please provide data flow graphs of above code like figure 5.13, figure 5.14 and figure 5.15 in ICS book. Please don't just provide final graph, you need do it one by one (from figure 5.13 to figure 5.15) and bold the critical path on the final graph. (13')