ICS QUIZ 1

March 9, 2022

1 Y86 (Organization)

Read the following assembly of Y86,and answer the questions. Note that we introduce a new intruction **modq**.

```
0x000:                          |.pos 0
0x000:                          | init:
0x000: ___[1]___                |     irmovq stack, %rsp
0x00a: 80 2800000000            |           call main
0x013: 00                       |     halt
                                |
0x018:                          |.align 8
0x018: 32000000 00000000        | array: .quad 0x32
0x020: 24000000 00000000        |        .quad 0x24
                                |
0x028:                          | main:
0x028: 30f2 18000000 00000000   |     ___[2]___
0x032: 5027 00000000 00000000   |     mrmovq (%rdx), %rdi
0x03c: ___[3]___                |     mrmovq 8(%rdx), %rsi
0x046: 80 50000000 00000000     |     call gcd
0x04f: 90                       |     ret
                                |
0x050:                          | gcd:
0x050: ___[X]___                |    modq %rsi, %rdi /* %rdi = %rdi mod %rsi */
0x052: 30f1 00000000 00000000   |    irmovq $0, %rcx
0x05c: ___[4]___                |    subq %rcx, %rdi
0x05e: 74 6a000000 00000000     |    ___[5]___
0x067: 2060                     |    rrmovq %rsi, %rax
0x069: 90                       |    ret
0x06a:                          | swap:
0x06a: 2060                     |    rrmovq %rsi, %rax
0x06c: 2076                     |    rrmovq %rdi, %rsi
0x06e: 2007                     |    rrmovq %rax, %rdi
0x070: 80 50000000 00000000     |    call gcd
0x079: 90                       |    ret
                                |


___[6]___:                      |.align 8
...                             | stack:
...                             |       ...
```

1.1

The **modq** instruction is a new member of **OPq** type. It compute rB mod rA and write to rB, and the CC is not affected. Assume the fn code of **modq** is 4. To implement **modq** in sequential Y86-64, we have to make some modification to the hardware.
Which component(s) of sequential Y86-64 implementation should be modified ? ( )
A. change fetch stage logic for this new instruction
B. change ALU to add mod logic.
C. add a new connection wire to pass the remainder(rB % rA).
D. change the update PC logic

1.2

Please fill in the blanks within above Y86-64 binary and assembly code. (use label name if possible)

Blank	answer (binary / assembly)
X	
1	
2	
3	
4	
5	
6	

1.3

Suppose the entry point is *init*, please calculate the value of %rax after the program HALT.

2 Exception (System)

2.1 True or False

1) Operating System is a program that manages the all computer hardware.

2) Application program must use operating system to access the computer hardware resource.

3) There are three fundamental abstractions provided by operating system: processor, virtual memory and file

4) Operating system provides the illusion that the program is the only one running on the system.

5) The program appears to have exclusive use of all the processors, main memory, and I/O devices

6) There are two exclusive parts in a process: user code and OS code

7) All devices are modeled as files

2.2 Short question

1) Can a process run on physical memory directly. If it can, please compare it with running on virtual memory; if not, give your reason.

2) How many parameters can be passed in a system call (limited or unlimited). Can parameters be passed in the stack like normal function call, why?

3) Please draw the control flow transferring graph for this procedure: a time interrupt comes when kernel is disposing a system call invoked by user program

4) How does kernel handle the exception using the exception table? Use time interrupt as an example (Exception number of time interrupt is 30)

2.3 Comprehensive Question

Bob writes the following function and uses the **objdump** to get its memory layout in a X86_64 machine. Assume the time interrupt frequency is 1000/s. Please show all possible exceptions during the execution. Please answer the question using the following format (Line, kind of exception, asynchronous or synchronous, definitely or not, reason)

Address	Offset	perm	size	mapping
0x400000	0	r-x	0x3000	.text
0x403000	0x1000	rw-	0x1000	.data
0x600000	0	rw-	-	.heap
0x7fffffde000	0	rw	-	.stack

```
1 void func1(void) {
2     char *ptr = 0x401000;
3     char ptr_val = *ptr;
4     char *ptr2 = malloc(0x1000);
5     char *ptr3 = 0x404500;
6     int c;
```

```
    c = getchar();
    *ptr3 = 1;
    *ptr = 1;
    return 0;
}
```

3 Pipeline (Organization)

3.1 Short Question

All the questions in this section are based on the standard pipeline implementation from Figure 4.52 in CSAPP.

1) Is it possible to forward the m_valM to ALU_A or ALU_B to solve Load Use hazard? If it is possible, why not forwarding the value in the textbook? If not, please give your explanation.

2) What are programmer-visible states?

3) What would happen if the pipeline read an invalid instruction? (**Hint**: Consider M/W stage pipeline registers.)

4) If we take the following branch prediction policy, how to modify the f_predPC logic?

```
target address < current PC ? taken : not taken
```

5) Is it true that the more pipeline stages, the faster programs run? Why? Please give your explanations.

3.2 Comprehensive Question

Bob has a computer with a **single memory** and wants to implement a pipeline from Figure 4.52 in CSAPP book. Thus bob decides to modify the hardware logic. The modified pipeline is shown in the end of the paper. **NOTE:** A single memory means that **only one memory access(read or write) is allowed within a clock cycle**.

1) Please fill the blanks below to implement the pipeline.

```
word mem_addr = [
    M_icode in  IRMMOVQ, IPUSHQ, ICALL, IMRMOVQ  : M_valE;
    M_icode in  IPOPQ, IRET  : M_valA;
     [1]  :  [2]
];
bool mem_read =  [3] ;
bool mem_write =  [4] ;
```

2) Because Bob only has one memory, there will be some hazards when fetching instruction and accessing memory simultaneously. Please list new detect conditions like Figure 4.64 and new control actions like Figure 4.66. Use "–" for no additional control action("normal").

Condition	Trigger				
Structure Hazard	[1]				
	Pipeline register				
	F	D	E	M	W
	[2]	[3]	[4]	[5]	–

3) The new structure hazard introduced some combination of hazards. How many **new combinations of hazards** are there with the structure hazard? And we need to modify the pipeline register control logic, please write down the modified HCL implementation for pipeline register.

4) Bob runs the following y86-64 code with his pipeline implementation, please calculate total cycles and CPI respectively. (**NOTE:** you need calculate the number of cycles until the last stage of the last instruction.)

```
# copy elements from src to dst
main:                       loop:
    irmovq  src, %rdi           mrmovq (%rdi), %rax
    irmovq  dst, %rsi           rmmovq %rax,   (%rsi)
    irmovq  $0x3, %rdx          iaddq  $-1,    %rdx
    call    copy                iaddq  $8, %rdi
    halt                        iaddq  $8, %rsi
copy:                           jmp     copy
    andq    %rdx, %rdx  L1:
    jle     L1                  ret
```

Total cycles for current implementation is [1] , CPI is [2] .

4 Signal(System)

4.1 Short Question

```
#include "csapp.h"

volatile sig_atomic_t flag = 0;

void handler(int s)
{
```

```
7       flag = s;
8   }
9
10  int main(int argc, char **argv)
11  {
12      Signal(SIGUSR1, handler);
13      pid_t pid = getpid();
14
15      if (Fork() == 0) {
16          Kill(pid, SIGUSR1);
17          exit(0);
18      }
19
20      flag = 0;
21      while (!flag) {
22
23      }
24
25      printf("flag: %d\n", flag);
26      printf("exiting...\n");
27      exit(0);
28  }
```

1) Sometimes this program will run forever. Provide a possible execution flow of the program which will run forever. Describe it in Chinese or English, or by drawing a simple graph.

2) Can you fix the bug by adding some lines of code. Hint: You will get full mark if you correctly use **sigsuspend**. Otherwise, a busy polling version is also acceptable.

3) Can we remove the key word **volatile**? Why?

4) What is the final value of **flag** if this program successfully exits?

4.2 Comprehensive Question

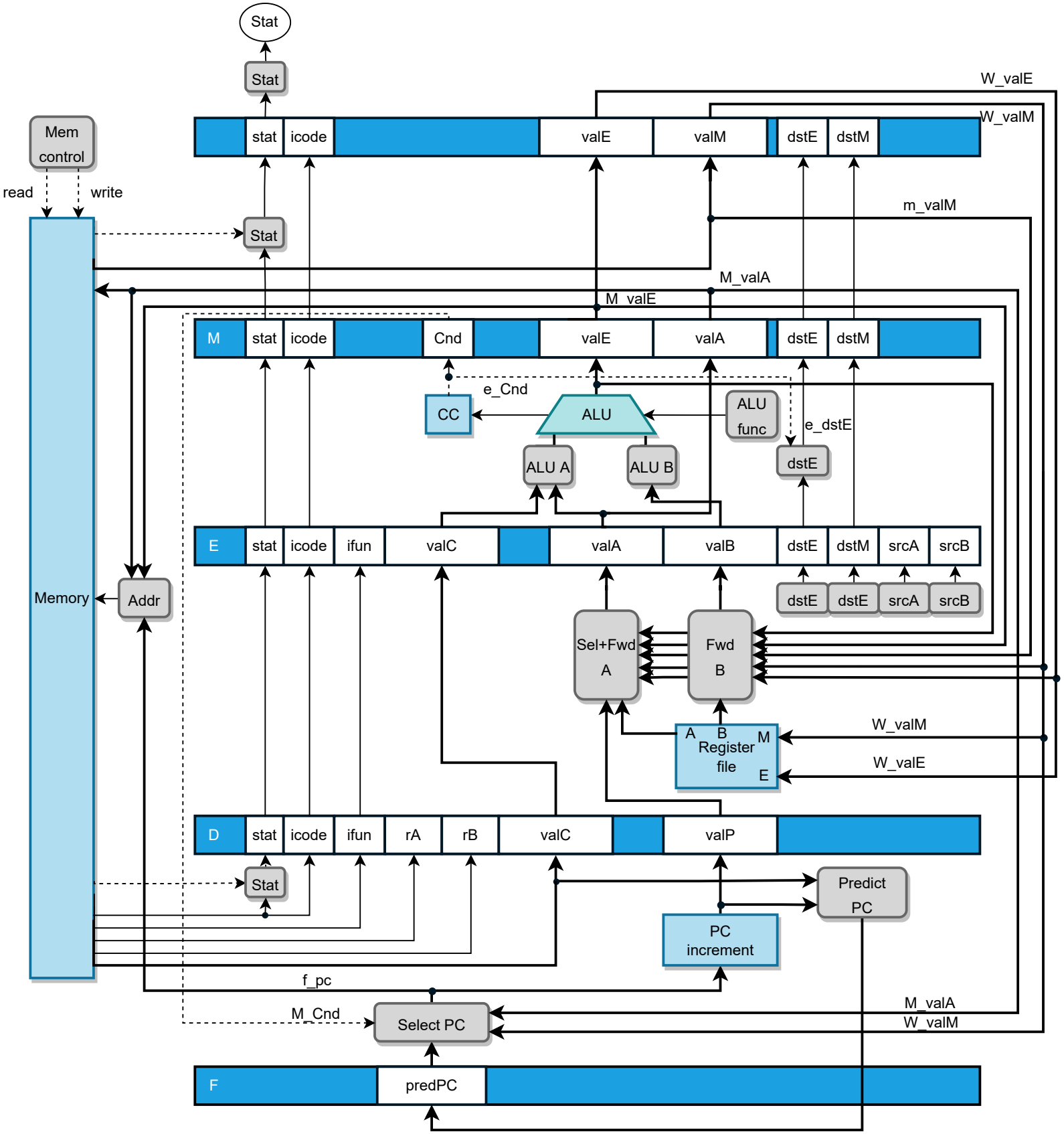
```
1   #include "csapp.h"
2
3   #define MAX 20
4
5   volatile sig_atomic_t number = 2;
6   volatile pid_t parent;
7   volatile pid_t child;
8
9   volatile char array[MAX-1];
```

```
10
11 void sigusr1_handler(int s)
12 {
13     int finish = 1;
14     Sio_putl(number);
15     Sio_puts(" ");
16
17     for (int i = number+1; i <= MAX; i++) {
18         if (i % number == 0) {
19             array[i-2] = 1;
20         }
21     }
22
23     for (int i = number+1; i <= MAX; i++) {
24         if (array[i-2] == 0) {
25             number = i;
26             finish = 0;
27             break;
28         }
29     }
30
31     if (finish) {
32         Sio_puts("\n");
33         Kill(child, SIGINT);
34     } else {
35         Kill(child, SIGUSR2);
36     }
37 }
38
39 void sigusr2_handler(int s)
40 {
41     Kill(parent, SIGUSR1);
42 }
43
44 void sigint_handler(int s)
45 {
46     Kill(parent, SIGKILL);
47     _exit(0);
48 }
49
50 int main(int argc, char **argv)
51 {
52     Signal(SIGUSR1, sigusr1_handler);
53     Signal(SIGUSR2, sigusr2_handler);
54     Signal(SIGINT, sigint_handler);
55
```

```
56      memset((void*) array, 0, sizeof(array));
57
58      parent = getpid();
59      if ((child = Fork()) == 0) {
60          Kill(parent, SIGUSR1);
61      }
62
63      while (1) {
64          Sleep(1);
65      }
66
67      exit(0);
68  }
```

1) We cannot call functions like **printf** or **exit** in the signal handler. Why?

2) Read the program and give the output of the program.


Stat
Stat
W_valE
W_valM
Mem
control
read
write
stat
icode
valE
valM
dstE
dstM
m_valM
Stat
M_valA
M_valE
M
stat
icode
Cnd
valE
valA
dstE
dstM
e_Cnd
CC
ALU
ALU
func
e_dstE
ALU A
ALU B
dstE
E
stat
icode
ifun
valC
valA
valB
dstE
dstM
srcA
srcB
dstE
dstE
srcA
srcB
Memory
Addr
Sel+Fwd
A
Fwd
B
A
B
M
Register
file
E
W_valM
W_valE
D
stat
icode
ifun
rA
rB
valC
valP
Stat
Predict
PC
PC
increment
f_pc
M_Cnd
Select PC
M_valA
W_valM
F
predPC