ICS Homework 4

March 28, 2020

1 Organization

1.1 Hazard

```
# demo.ys
0x000:     irmovq stack,%rsp
0x00a:     call p
0x013:     irmovq $5,%rsi
0x01d:     halt

0x020:.pos 0x20
0x020:p:   irmovq $-1,%rdi
0x02a:     ret  #below will not be executed
0x02b:     irmovq $1,%rax
0x035:     irmovq $2,%rcx
0x03f:     irmovq $3,%rdx
0x049:     irmovq $4,%rbx

0x100:.pos 0x100
0x100:stack:
```

1. During executing the above example, how many hazards will happen? Please point them out.

2. How could the above data hazards be handled? Please describe in detail.

3. What is the difference between **stall** and **bubble**?

1.2 Control Combination

1.2.1

Write a Y86-64 assembly-language program that causes combination A (Figure 4.67 in ICSAPP) to arise and determines whether the control logic handles it correctly.

1.2.2

Write a Y86-64 assembly-language program that causes combination B (Figure 4.67 in ICSAPP) to arise and completes with a halt instruction if the pipeline operates correctly.

2 System Software

2.1 Signal handler

One TA writes the following code about a user-defined signal handler on a x86-64 Linux machine. Read this C program and answer the questions below.

```c
#include <signal.h>
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

void handler(int sig)
{
        printf("hello\n");
}

int main(void)
{
        signal(SIGINT, handler);
        kill(getpid(), SIGINT);
        while(1);
        return 0;
}
```

1. Does the function ‘handler’ run in user mode or kernel mode?
2. What’s the output of this program? When we type a ‘ctrl-c’, what will happen? How to use *kill* command to stop this program?
3. When we run this program in gdb, can we see the same result as Q2? If not, explain why and show how to fix it to get the same result in gdb as Q2.