Homework 10

1. PIPELINE EXCEPTION HANDLING

   You are required to extend the PIPE implementation so that it will reboot after any exception. The CPU should only reboot when the exception reaches the W stage. After the reboot PC is reset back to 0 and other registers, CC and memory should be preserved. For simplicity, you *DO NOT* need to consider the combination between reboot hazard and other hazards.

   (a) What is the `f_pc` and `W_stall` now?

   (b) How many other instructions have entered the pipeline when the first instruction after reboot is in the FETCH stage?

   (c) How do you prevent these instructions (instructions that *AFTER* the exception and *BEFORE* the first instruction after reboot) from modifying registers, CC and memory?

2. MACHINE INDEPENDENT OPTIMIZATION

   Suppose we have some codes as below.

```
typedef struct {
        int vals[3];
} block_t;

typedef struct {
        int length;
        block_t *blocks;
} blocklist;

int get_length(blocklist *bl)
{
```

```
        return bl->length;
}

block_t* get_blocks(blocklist *bl)
{
        return bl->blocks;
}

void SUM(blocklist *bl, long *dest)
{
        for (int i = 0; i < get_length(bl); i++) {
                int size = 1;
                for (int j = 0; j < 3; j++)
                        size = size * get_blocks(bl)[i].vals[j];
                *dest = *dest + size;
    }
}
```

Try to optimize the function `SUM` with a combination of optimizations you have learned in the ICS class. Comment briefly on your optimizations.