ICS Homework 5

March 23, 2021

1 Pipeline

Suppose we add a new instruction *rjmp rB* to Y86 instruction sets. It will jmp to the address stored in *rB*.
1. Fill in the function of each stage for *rjmp rB* instructionin Y86 sequential implementation. (NOTE: use *valB* to update PC)

Stage	*rjmp rB*
F	icode:ifun ¡- M1[PC] rA:rB ¡- M1[PC+1] valP ¡- PC + 2
D	valB ¡- R[rB]
E	–
M	–
W	–
P	PC ¡- valB

2. As shown in the new PIPE logic figure, we add a forwarding logic from *E_valB* to *f_pc* to support *rjmp* instruction, since the target address require read from register file. Please describe all possible hazards due to new instruction *rjmp*. You need provide detail explanation and list detection conditions like Figure 4.64 and control action like Figure 4.66. (Do not consider the hazard combinations here.)

Condition	Trigger
Target-addr Hazard	IRJMP in {D_icode}

F	D	E	M	W
Stall	Bubble	Normal	Normal	Normal

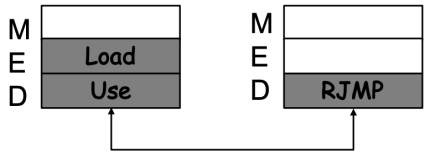
Condition	Trigger
Processing `ret`	IRET ∈ {D_icode, E_icode, M_icode}
Load/use hazard	E_icode ∈ {IMRMOVQ, IPOPQ} && E_dstM ∈ {d_srcA, d_srcB}
Mispredicted branch	E_icode = IJXX && !e_Cnd
Exception	m_stat ∈ {SADR, SINS, SHLT} || W_stat ∈ {SADR, SINS, SHLT}

Figure 4.64 **Detection conditions for pipeline control logic.** Four different conditions require altering the pipeline flow by either stalling the pipeline or canceling partially executed instructions.

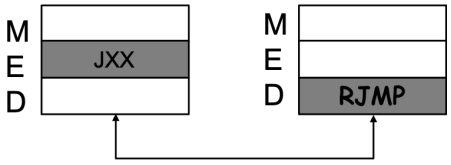
	Pipeline register				
Condition	F	D	E	M	W
Processing `ret`	stall	bubble	normal	normal	normal
Load/use hazard	stall	stall	bubble	normal	normal
Mispredicted branch	normal	bubble	bubble	normal	normal

Figure 4.66 **Actions for pipeline control logic.** The different conditions require altering the pipeline flow by either stalling the pipeline or canceling partially executed instructions.

3. Please list all hazard combinations (arise simultaneously) including *rjmp* instruction. You need draw pipeline states figures like Figure 4.67 and list pipeline control action like tables after Figure 4.67 for each combination.



Condition	F	D	E	M	W
Load	Stall	Stall	Bubble	Normal	Normal
RJMP	Stall	Bubble	Normal	Normal	Normal
Combination	Stall	B+S	Bubble	Normal	Normal
Desired	Stall	Stall	Bubble	Normal	Normal



Condition	F	D	E	M	W
JXX	Normal	Bubble	Bubble	Normal	Normal
RJMP	Stall	Bubble	Normal	Normal	Normal
Desired	S or B or N	Bubble	Bubble	Normal	Normal

4.The original PIPE implementation of Y86 should be modified to support the *rjmp* instruction. Please describe the modification and provide HCL of *f_pc* and *D_bubble* logic. (NOTE: Only need to write the code about *rjmp* instruction)

```
word f_pc = [
    E_icode == IRJMP : E_valB
];


bool D_bubble = [
    D_icode == IRJMP && !{E_icode in { IMRMOVQ, IPOPQ } &&
E_dstM in { d_srcA, d_srcB}}
];
```

2 Signal

```
1 int counter = 2;
2
3 void handler1(int sig) {
4     counter = counter + 1;
```

```
5       printf("%d\n", counter);
6       exit(0);
7   }
8
9   int main() {
10      signal(SIGINT, handler1);
11      printf("%d\n", counter);
12      if ((pid = fork()) == 0) {
13          while(1) {};
14      }
15      kill(pid, SIGINT);
16
17      counter = counter -1;
18      printf("%d\n", counter);
19      waitpid(-1, NULL, 0);
20      counter = counter + 1;
21      printf("%d\n", counter);
22      exit(0);
23  }
```

1. Please rewrite the *handler* according to the guidelines in section 8.5.5 (HINT: you can use *Sio_puts* as thread safe *printf* if needed).

```
1   volatile sig_atomic_t counter = 2;
2
3   void handler1(int sig) {
4       int olderrorno = errorno;
5       sigset_t mask, prev_mask;
6       Sigfillset(&mask);
7       Sigprocmask(SIG_BLOCK, &mask, &prev_mask);
8       counter = counter + 1;
9       Sio_putf("%d\n", counter);
10      Sigprocmask(SIG_BLOCK, &prev_mask, NULL);
11      errorno = olderrorno;
12      _exit(0);
13  }
14
15  int main {
16      .........
```

2. Please write down all the possible outputs of the original programs.
   2\n3\n1\n2\n or 2\n1\n3\n2\n