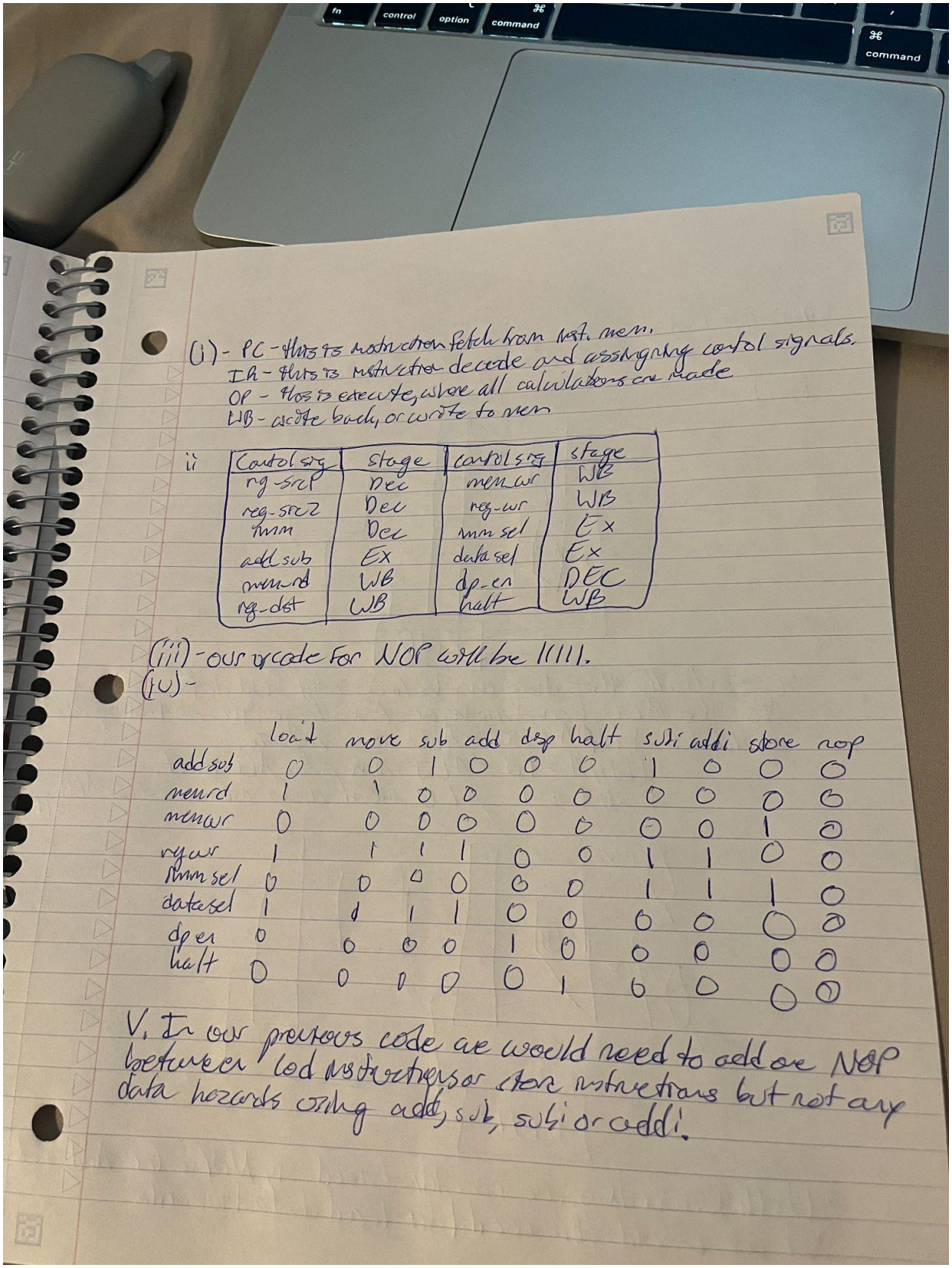
Code for our working L7

```
2000 // sub r0, r0, r0
2444 // sub r4 r4 r4
2555 // sub r5 r5 r5
7006 //addi r0 r0 6
7446
7556
3000 // add r0 r0 r0
3444
3555
3000 // add r0 r0 r0
3444
3555
3000 // add r0 r0 r0
3444 //add r4 r4 r4
3555 //add r5 r5 r5
0100 // load r1, r0, 0
7441 //addi r4 r4 1
4010 // disp r1, 0, 0
7552 //addi r5 r5 2
0240 //load r2 r4 0
4020 //disp r2
0350 //load r3 r5 0
3612 //add r6 r1, r2
f000
f000
8036 //store r3, r6
f000
f000
3763 //add r7 r6 r3
f000
f000
4070
```

5000 // HALT



(i) - PC - this is instruction fetch from inst. mem.
IR - this is instruction decode and assigning control signals.
OP - this is execute, where all calculations are made
WB - write back, or write to mem

ii

Control sig	stage	control sig	stage
rg-src1	Dec	mem_wr	WB
reg-src2	Dec	reg_wr	WB
imm	Dec	imm sel	Ex
add_sub	Ex	data sel	Ex
mem_rd	WB	dp_en	DEC
rg_dst	WB	halt	WB

(iii) - our opcode for NOP will be 11111.

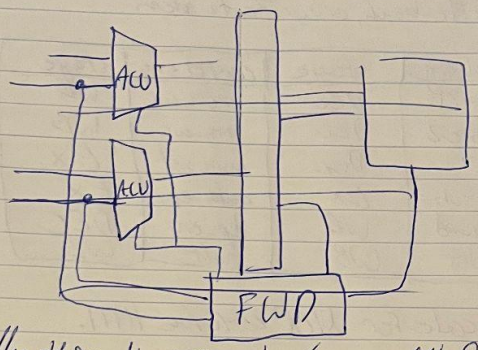
(iv) -

	load	move	sub	add	disp	halt	subi	addi	store	nop
add sub	0	0	1	0	0	0	1	0	0	0
mem rd	1	1	0	0	0	0	0	0	0	0
mem wr	0	0	0	0	0	0	0	0	1	0
reg wr	1	1	1	1	0	0	1	1	0	0
imm sel	0	0	0	0	0	0	1	1	1	0
data sel	1	1	1	1	0	0	0	0	0	0
dp en	0	0	0	0	1	0	0	0	0	0
halt	0	0	0	0	0	1	0	0	0	0

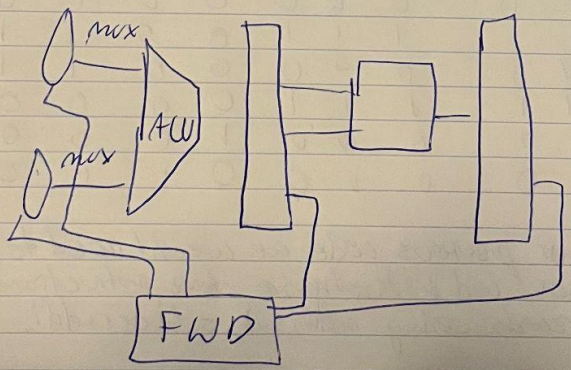
V. In our previous code we would need to add one NOP between lod instructions or store instructions but not any data hazards using add, sub, subi or addi.

(ii)(i)- We had no problems implementing the system nor did we run into any issues while designing.

(iii)-



- Essentially this diagram below will send data to the ALU from mem or Ex stage. The control signals would be FWDA = 00 - ALU from reg file.
  FWDA = 10 - ALU from prior ALU result.
  FWDA = 01 - ALU from data memor earlier ALU result.



- With forwarding we wouldn't need any NOP's for Add, sub but we would still need 1 for load data hazards.