Ethan West
CS 3339.001
Homework 4

1) a. R-type: (add, sub, and, or) = 5ns + 4ns + 2ns + 6ns + 2ns + 4ns = 23ns
jump = 5ns + 2ns = 7ns
beq = 5ns + 4ns + 2ns + 6ns + 2ns + 2ns = 21ns
lw = 5ns + 4ns + 2ns + 6ns + 5ns + 2ns + 4ns = 28ns
sw = 5ns + 2ns + 6ns + 5ns = 22ns

b.

Control Signal	add	sub	and	or	jump	beq	lw	sw
Reg Dst	1	1	1	1	0	0	0	1
Jump	0	0	0	0	1	0	0	0
Branch	0	0	0	0	0	1	0	0
Mem Read	0	0	0	0	0	0	1	0
Mem to Reg	0	0	0	0	0	0	1	0
Mem Write	0	0	0	0	0	0	0	1
ALU src	0	0	0	0	0	0	1	1
Reg write	1	1	1	1	0	0	1	0
ALU op (2 bits)	10	10	10	10	-	01	00	00
ALU operation	add	sub	and	or	-	sub	add	add

2) a.

```
addi $s3, $s1, 5
lw $s2, 4($s3)      $s3, requires forwarding/stalling
lw $s4, 3($s1)
add $s2, $s3, $s2   $s2, requires forwarding & $s3, requires forwarding
sw $s2, 0($s3)      $s3, requires forwarding
```

b.

```
IF ID EX ME WB
   NOP
      NOP
         IF ID EX ME WB
            IF ID EX ME WB
               NOP
                  IF ID EX ME WB
                     NOP
                        NOP
                           IF ID EX ME WB
```

```
addi $s3, $s1, 5
NOP
NOP
lw $s2, 4($s3)
lw $s4, 3($s1)
NOP
add $s2, $s3, $s2
NOP
NOP
sw $s2, 0$(s3)
```

2) c. It is not possible to reduce the total number of NOPs any further

d. The program will execute properly as the hazard detection unit only inserts stalls which are not necessary for this program

e.

Cycle	1	2	3	4	5	6	7
addi $s3, $s1, 5	IF	ID	EX	ME	WB		
lw $s2, 4($s3)		IF	ID	EX	ME	WB	
lw $s4, 3($s1)			IF	ID	EX	ME	WB
add $s2, $s3, $s2				IF	ID	EX	ME WB
Forward A	00	00	00	10	00	00	00
Forward B	00	00	00	00	00	01	00

IF/ID Write & PC Write are always 1 because there are no NOPs.
ID/EX Mux is always 01 (lets control values pass)

f. The value of the destination register from EX/MEM & MEM/WB is needed to detect the data hazards between addi & lw and lw & add respectively.

g.

Cycle	1	2	3	4	5
addi $s3, $s1, 5	IF	ID	EX	ME	WB
lw $s2, 4($s3)		IF	ID	-	-
lw $s4, 3($s1)			IF	-	-
PC Write	1	1	1	0	0
IF/ID Write	1	1	1	0	0
ID/EX MUX	0	0	0	1	1

3) a. Offset = 5 bits → Block size = 2^5 = 32 bytes → Block size = $\frac{32}{4}$ =

8 4-byte words

b. index = 5 → # of entries = 5^2 = 32 lines

c. Cache Stores = 32 lines · 8 words per block · 4 bytes per word =

1024 bytes = 8192 bits

Cache Uses = 8192 + (22 + 1) 32 = 8928

Ratio = $8928/8192$ = 1.09

d.

Byte address					
Hexa	T	I	O	Hit / Miss	Bytes Replaced
0x00	00	00000	00000	Miss	
0x04	00	00000	00100	Hit	
0x10	00	00000	10000	Hit	
0x84	00	00100	00100	Miss	
0xE8	00	00111	01000	Miss	
0xA0	00	00101	00000	Miss	
0x400	01	00000	00000	Miss	Tag = 0x00
0x1E	00	00000	11110	Miss	Tag = 0x01
0xBC	00	00101	10100	Hit	
0xC1C	11	00000	11100	Miss	
0xB4	00	00101	10100	Hit	Tag = 0x00
0x884	10	00100	00100	Miss	Tag = 0x00

e. Hit Ratio = 4/12 = 1/3 = .33

f. <0x00, 0x03, C1C>
<0x04, 0x02, 884>
<0x05, 0x00, B4>
<0x07, 0x00, E8>