

# Solution Guide for Midterm Exam, Spring, 1999

## 1. Machine code and memory accesses for *Stack* and *Register*.

In both cases we will omit the code for checking that the initial value of *i* in the loop satisfies the  $i \leq 10$  condition, which a smart compiler would figure out anyway...

Points were 5 each for (a) & (b), 2 each for (c)(a) & (c)(b)

### (a) *Stack*:

label	Op	Opnd	Comment	Mem Ref	Times
	PUSH	ZERO		8	1
	POP	SUM	sum = 0	8	1
	PUSH	ONE		8	1
	POP	I	i = 1	8	1
	PUSH	ONE	for required POP below	8	1
LOOP	POP	JUNK	unload stack	8	10
	PUSH	I		8	10
	PUSH	SUM		8	10
	ADD			1	10
	POP	SUM	sum = sum + i	8	10
	PUSH	I		8	10
	PUSH	ONE		8	10
	ADD			1	10
	POP	I	i = i + 1	8	10
	PUSH	I		8	10
	PUSH	TEN		8	10
	SUB			1	10
	JN	LOOP	jump if i <= 10	4	10
	POP	JUNK	clean out stack	8	1
	HALT			1	1
ZERO	.DW	0			
ONE	.DW	1			
TEN	.DW	10			
JUNK	.DW	99	SO, number of memory references will be:		
SUM	.DW	99			
I	.DW	99	(79 * 10) + 49 or <u>839</u>		

(Note — memory bus only 8 bits wide,  
hence instruction + operand takes 8 references)

Note that JN LOOP only takes 4

(b) **Register:**

label	Op	Opnd	Comment	Mem Ref	Times
	LDA	ZERO		8	1
	STA	SUM	sum = 0	8	1
	LDA	ONE		8	1
	STA	I	i = 1	8	1
LOOP	LDA	I		8	10
	ADD	SUM		8	10
	STA	SUM	sum = sum + i	8	10
	LDA	I		8	10
	ADD	ONE		8	10
	STA	I	i = i + 1	8	10
	SUB	TEN		8	10
	JN	LOOP	jump if i <= 10	4	10
	HALT	1234H		4	1
ZERO	.DW	0			
ONE	.DW	1			
TEN	.DW	10			
JUNK	.DW	99	SO, number of memory references will be:		
SUM	.DW	99			
I	.DW	99	(60 * 10) + 36 or <u>636</u>		

2. This simply involves copying from the text for SRC, page 66, but substituting the SPARC notation. Many people failed to specify how the shift count is determined:

$n := \text{opnd2} \langle 4..0 \rangle$

3(a) 12500 seconds, which is 3 hours, 28 minutes, 20 seconds.

(b) Approximately 4,800 floppy disks.

4. An almost identical question, with different data values, was on last year's midterm, to which you all had access. So, you should have been prepared for a question like this.....

(i) **0B940008** (5)

(a) **ld r14, 8(10)**

(b) **register 14 and PC**

(c) **r14 = 00000004, PC = 00000104** (add 1 if got the PC change in (i) & (ii))

(ii) 71954000 (5)

(a) *sub r6, r10, r20*

(b) *register 6 and PC*

(c) *r6 = FFFFFFFC, PC = 00000104*

(iii) 4FE8A003 (8)

(a) *brlnz r31, r20, r10*

(b) *register 31 and PC*

(c) *r31 = 00000104, PC = 0000000C*

5.

```
DA
87  1101 1010 1000 0111      LOOP: ADD.L D7, D5
                                   (D5 is dst)
51
CB  0101 0001 1100 1011      DBF D3, LOOP
                                   -4
FF
FC  minus 4                    4 points for assembly code
```

XX

*PC: 1006<sub>16</sub> 00001006 (1008 if interpret "execute" literally - OK)*  
*(3 points)*

*D3: -1 FFFFFFFF*  
*(2 points)*

*D5: 6 00000006*  
*(3 points)*

*D7: 1 00000001*  
*(2 points)*

**(total of 10 points for the final register values)**

6. We'll discuss this question in class. The fetch-execute cycle is a fundamental idea in computer architecture, and a dismayingly large number of students cannot draw a comprehensible and correct picture of this cycle. I'll also talk about interrupts and pipelined machines.