Princeton University
COS 217: Introduction to Programming Systems
Fall 2001 Midterm Exam 2 Answers

Students were allowed to use the Paul textbook during the exam.

Question 1

Flattened C program:

```
   i = 0;
loop:
   if (i >= (n-1)) goto loopend;
   if (items[i] <= items[i+1] goto ifend;
   swap(items, i);
ifend:
   ++i;
   goto loop;
loopend:
```

Assembly language program:

```
! Register map:
! %l0  int i
! %l1  int *items
! %l2  int n;

   ! i = 0;
   clr  %l0

loop:

   ! if (i >= (n-1)) goto loopend;
   sub  %l2, 1, %l3
   cmp  %l0, %l3
   bge  loopend
   nop

   ! if (items[i] <= items[i+1]) goto ifend;
   set  items, %l3
   sll  %l0, 2, %l4
   ld   [%l3 + %l4], %l5
   add  %l4, 4, %l4
   ld   [%l3 + %l4], %l6
   cmp  %l5, %l6
   ble  ifend
   nop

   ! swap(items, i);
   set  items, %o0
   mov  %l0, %o1
   call swap
   nop

ifend:

   ! ++i;
   inc  %l0

   ! goto loop;
   ba   loop
   nop

loopend:
```

Question 2

Leaf subroutines are easier to implement and more efficient because they avoid executing the save and restore instructions.

Leaf subroutines are allowed to alter the contents of the %o0 - %o5 registers and the %g registers. For SPARC ABI compliance, leaf subroutines should not alter the contents of the %g registers.

Question 3

```
#define count %l0
#define total %l1
#define value %o0
   ...
   clr count
   ba  test
   clr total
loop:
   inc count
test:
   call get_int
   nop
   cmp  value, 0
   bge,a loop
   add total, value, total
```

Question 4
---

(1) The get_info function assigns name_buf to current_account->name. But name_buf is on the stack, and so does not exist after get_info is finished executing. current_account->name thus becomes a dangling pointer.

(2) There is an error in both calls to malloc. Since accounts is a pointer, the expression sizeof(accounts) always evaluates to 4. The correct expression is sizeof(struct Account) or sizeof(*accounts).

(3) When the get_accounts function allocates new memory for the accounts array, it erroneously does not free the old array. That is a memory leak.

(4) When the get_accounts function allocates new memory for the accounts array, it does not change the value of current_account. Thus current_account erroneously refers to an address within the old array.

Question 5

Data Section

Offset	Contents (hex)	Explanation
0	61	.ascii "abc"
1	62	
2	63	
3	00	.skip 4

```
| 4      | 00              |
| 5      | 00              |
| 6      | 00              |
| 7      | 62              | .asciz "b"
| 8      | 00              |
| 9      | 04              | .byte 4
| 10     | 00              | .align 4
| 11     | 00              |
| 12     | 00              | .word 4
| 13     | 00              |
| 14     | 00              |
| 15     | 04              |
|--------|-----------------|
```

Text Section

Offset	Contents (hex)	Explanation
0	86 00 40 02	10 00011 000000 00001 0 00000000 00010
4	86 20 40 02	10 00011 000100 00001 0 00000000 00010
8	86 00 60 04	10 00011 000000 00001 1 0000000000100
12	86 00 60 08	10 00011 000000 00001 1 0000000001000
16	07 00 00 00*	00 00011 100 00000000000000000000
20	14 80 00 04	00 0 1010 010 00000000000000000100
24	40 00 00 00*	01 00000000000000000000000000000000
28	c6 00 40 02	11 00011 000000 00001 0 00000000 00010
32	c6 00 60 04	11 00011 000000 00001 1 0000000000100
36	c6 00 40 00	11 00011 000000 00001 0 00000000 00000

* Cannot be fully assembled

Question 6

The instruction set would need to be changed such that 6 bits (instead of 5) are used to denote each register operand.

Question 7

(a) 96 bytes.

Explanation:

```
 92 bytes (always required)
+ 4 bytes (7th actual parameter to f)
---
 96 bytes
+ 0 bytes (must be divisible by 8)
---
 96 bytes
```

(b) `save %sp, (-92 - 8) & -8, %sp`
or
`save %sp, -104, %sp`

Explanation:

```
 92 bytes (always required)
+ 4 bytes (local variable b)
+ 4 bytes (7th actual parameter to f)
---
100 bytes
+ 4 bytes (must be divisible by 8)
---
```

```
        104 bytes
```

(c) 512 bytes

     Explanation:

```
         96 bytes (for main)
        104 bytes (for f, where a6 == 3)
        104 bytes (for f, where a6 == 2)
        104 bytes (for f, where a6 == 1)
      + 104 bytes (for f, where a6 == 0)
      -----
        512 bytes
```

(d) ld [%fp + 92], %l0
     st %l0, [%fp - 4]

     Explanation:

     a6, a parameter, is stored in the caller's stack frame at %fp+92.
     b, a local variable, is stored in the current function's stack
     frame at %fp-4.

     Note: [%sp+96] is not the same as %[fp-4]. The size of the
     stack frame of subroutine f is 104 due to the alignment, and there
     is a 4-byte gap between the actual parameters and the local
     variables. If there were no such gap, then [%sp+96] would be the
     same as [%fp-4].

Question 8