

CSE 320 Sample Exam #2

1. (12 pts) Consider the following SPARC assembly language statements. For each statement which contains a comment, give the requested information after the instruction is executed. Give the value of the register in hexadecimal and the value of the condition code bits in binary.

```
set    0xc90000e4, %g6
set    0xa50000b7, %g7

or     %g6, %g7, %10    ! %10:  ed0000f7

and    %g6, %g7, %11    ! %11:  810000a4

sub    %g6, %g7, %12    ! %12:  2400002d

addcc  %g6, %g7, %13    ! %13:  6e00019b
                        ! NZVC:  0011

sra    %g6, 8, %14      ! %14:  ffc90000

srl    %g6, 12, %15     ! %15:  000c9000

sll    %g6, 16, %16     ! %16:  00e40000
```

2. (10 pts) Use the SPARC microprocessor tables (on the supplementary sheet) to answer the following questions.

a) Consider the SPARC machine language instruction shown below in hexadecimal and binary. Circle and label each field within the binary representation of the instruction, then give a corresponding SPARC assembly language statement.

Machine language instruction: B6430010

Binary: 1 0 1 1 0 1 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

Assembly language statement: addx %r12, %r16, %r27

or: addx %o4, %l0, %i3

b) Consider the SPARC machine language instruction shown below in hexadecimal and binary. Circle and label each field within the binary representation of the instruction, then give a corresponding SPARC assembly language statement.

Machine language instruction: EC3FA004

Binary: 1 1 1 0 1 1 0 0 0 0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0

Assembly language statement: std %r22, [%r30+4]

or: std %l6, [%i6+4]

3. (10 pts) For each of the following SPARC assembly language statements, give the corresponding SPARC machine language instruction in binary and in hexadecimal. Show your work for possible partial credit.

a) Assembly language statement: sra %g2, 20, %o1

Binary: 1 0 0 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0

Machine language instruction: 9338a014

b) Assembly language statement: ldub [%i5+%l2], %l7

Binary: 1 1 1 0 1 1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0

Machine language instruction: ee0f4012

4. (8 pts) Give a SPARC assembly language code segment which is equivalent to the following C statements. Assume that all variables are integers, the value of "aaa" is in register %10, and the value of "bbb" is in register %11.

```
if (aaa < 100 || aaa > 200)
{
  bbb = aaa + 50;
}
else
{
  bbb += 10;
}
```

```
        cmp    %10, 100
        bl     then
        nop
        cmp    %10, 200
        bg     then
        nop
        add    %11, 10, %11
        ba     endif
        nop
then:
        add    %10, 50, %11
endif
```

5. (8 pts) Give a SPARC assembly language code segment which is equivalent to the following C statements. Assume that all variables are integers, the value of "ccc" is in register %12, and the value of "ddd" is in register %13.

```
ccc = 0;
for (ddd = 10; ddd < 100; ddd += 5)
{
  ccc = ccc + 10 * ddd;
}
```

```
        mov    0, %12
        mov    10, %13
loop:
        cmp    %13, 100
        bge   endloop
        nop
        smul   %13, 10, %17
        add    %12, %17, %12
        add    %13, 5, %13
        ba     loop
        nop
endloop:
```

6. (8 pts) When writing assembly language functions on the SPARC system, it is necessary for the programmer to use certain instructions and to obey certain conventions. Answer the following questions clearly but concisely.

a) When it is executed, where does the "call" instruction place its own address (the current value of the PC)? \_\_%o7\_(%r15)\_\_\_\_\_

b) When it is executed, how does the "ret" instruction compute its target address? \_\_%i7+8\_(%r31+8)\_\_\_\_\_

c) When a function begins executing, which instruction is used by that function to shift the register window? \_\_SAVE\_\_\_\_\_

d) When a function finishes executing, which instruction is used by that function to shift the register window? \_\_RESTORE\_\_\_\_\_

e) If a function is designed to be called by a C function and to return an integer value, where must the return value be placed? \_\_%i0\_(%r24)\_\_\_\_\_

f) Consider the SPARC assembly language code segment below, where the "call" and the "nop" are the only instructions between the two comments.

```
! *** Line 1 ***
```

```
    call sub
    nop
```

```
! *** Line 2 ***
```

Assuming that function "sub" obeys the SPARC subprogram conventions, which IU registers might be changed by calling "sub"? That is, list all IU registers which are not guaranteed to contain the same values at "Line 1" and "Line 2".

The globals (%g1-%g7)

The outs (%o0-%o7)

7. (6 pts) Consider the C source code statements shown below.

```
int test( int * A, int B )
{
    *A = 5 * B + 10;

    return B + 20;
}
```

Complete the SPARC assembly language code segment below so that the sequence of assembly language statements is equivalent to the C statements above.

```
        .global  test
        .section ".text"
        .align   4
test:
    save      %sp, -96, %sp
    smul     %i1, 5, %i7
    add      %i7, 10, %i7
    st       %i7, [%i0]
    add      %i1, 20, %i0
    ret
    restore
```

8. (10 pts) Consider the SPARC assembly language code segment shown below.

```

        .section ".data"
        .align 4
Loc1:
        .half 0x4444, 0x5555, 0x6666, 0x7777
        .half 0x8888, 0x9999, 0xaaaa, 0xbbbb
Loc2:
        .word 0x66666666, 0x77777777, 0x88888888
        .word 0x99999999, 0xAAAAAAAA, 0BBBBBBBB
        .skip 10
Loc3:

        .section ".text"
        .align 4
        set 0x89ABCDEF, %i0
        set Loc1, %i1
        set Loc2, %i2

        ld [%i2+0], %i0           !    10: 66666666
        ldsh [%i2+6], %i1        !    11: 00007777
        ldsh [%i2+6], %i2        !    12: 00007777
        ldsb [%i2+10], %i3       !    13: ffffffff88
        ldub [%i2+10], %i4       !    14: 00000088

        stb %i0, [%i1+2]
        sth %i0, [%i1+4]
        st %i0, [%i1+8]

```

a) Assume the address of "Loc1" is 0x00020120. Give the address of each of the symbols listed below as a four-byte hexadecimal value.

Loc2: 00020130

Loc3: 00020152

b) In the spaces provided next to the assembly language instructions, give the four-byte hexadecimal value in each of the indicated registers after the instructions execute.

c) In the memory dump below, give the hexadecimal value in each of the indicated memory locations after the instructions execute. You only need to give the value for memory locations which are changed by the instructions.

```

                ef      cd ef      89 ab cd ef
00020120: 44 44 55 55 66 66 77 77 88 88 99 99 aa aa bb bb
                --      -----

```

9. (6 pts) Consider the C source code statements shown below.

```
int examine( int, int * );

int A, B, C;

int main()
{
  C = examine( A, &B );
}
```

Complete the SPARC assembly language code segment below so that the sequence of assembly language statements is equivalent to the C statements above.

```
        .section ".data"
        .align 4
A:      .skip 4
B:      .skip 4
C:      .skip 4

        .global main
        .section ".text"
        .align 4
main:

        save    %sp, -96, %sp
        set    A, %l7
        ld     [%l7], %o0
        set    B, %o1
        call   examine
        nop
        set    C, %l7
        st     %o0, [%l7]
        ret
        restore
```

10. (12 pts) Consider the C source code statements shown below.

```
struct person
{
    char name[30];
    int id;
    int points;
};

char Fmt[] = "Name: %s ID: %d Points: %d\n";

void display_one( struct person List[], int I )
{
    printf( Fmt, List[I].name, List[I].id, List[I].points );
}
```

Complete the SPARC assembly language code segment below so that the sequence of assembly language statements is equivalent to the C statements above.

```
        .section ".data"
        .align 4
Fmt:    .asciz  "Name: %s ID: %d Points: %d\n"

        .global display_one
        .section ".text"
        .align 4
display_one:

        save    %sp, -96, %sp
        smul   %i1, 40, %l1
        add    %i0, %l1, %l0
        set    Fmt, %o0
        mov    %l0, %o1
        ld     [%l0+32], %o2
        ld     [%l0+36], %o3
        call   printf
        nop
        ret
        restore
```