Module 13 Example Set 3

1. Describe the effect produced by the following SparcV8 instruction:

```
save    %sp, -40, %sp
```

This instruction subtracts 40 from the stack point, thus reserving 10 words of storage on the stack. It also slides the register window to reveal a new set of logical registers. The output registers %o0 through %07 overlap with and correspond to registers %i0 through %i7 within the new register window.

2. How is a nop instruction implemented on the SparcV8?
The instruction sethi 0,%g0 is used as a nop. Since %g0 is hardwired to 0, this instruction has no effect other than consuming execution time.

3. The SparcV8 architecture includes base register plus indexed addressing. Does it also include base register plus index plus displacement? If not, why not.
Instructions that reference memory can employ a base register and an index register or they can use a base register plus a 13-bit signed displacement.
Bit 13 within the machine instruction = 0 when two registers are used in generating the memory operand address, bit 13 = 1 when one register plus the sign extended displacement is used as the operand address. The index register number is contained in the rightmost 5 bits of the machine instruction. The displacement is contained in the rightmost 13 bits of the instruction. So either one or the other can be used, both not both.

4. A leaf routine is one that is called but that calls no other function or routine. Leaf routines need not execute a Sparc "save" instruction. Show one or more instructions that return properly from a leaf routine which was invoked by the call instruction.
Since the return address is placed into the %o7 register by the call instruction, the leaf routine can perform a return to the caller by executing:
jmpl %o7,%g0 to jump back to the return address without saving anything.
(Recall that jmpl saves the return address in the destination. If the destination register is %g0, as in this case, the return address is discarded.)

5. To what memory address is control transferred in response to a reset trap on the SparcV8?
Resets cause a transfer to address 0.

6. On the SparcV8 a memory unaligned access trap is assigned trap number 7.
a) Write down a sequence of Sparc instructions that check for an unaligned access trap and branch to the instruction with label "unaligned" if the trap is type 7.
The trap base register can be read to determine the contents of the tt (trap type) field. For alignment traps, the 8-bit tt field will contain the value 7. The following instructions can be used:

```
rdtbr     %g2              read contents of TBR
xnorcc    %g2,0x70,%g2     result=0 if tt field = 7
be        unaligned
nop
```

b) When these instructions are executed, in what mode must the processor run?
Since rdtbr is a priviledge instruction, the processor must be in supervisor mode when the rdtbr is executed.

7. When a SparcV8 call instruction is executed, what are the contents of the pc and the npc?
The pc contains the address of the call instruction and npc contains the address of the function or procedure that is called.

8. Prior to executing the following SparcV8 instruction, register %g2 contains 0x4592BE8 and register %g4 contains 0x0E1E1000:

```
smul   %g2,%g4,%g6
```

Show, in hex, the contents of any registers that are modified by this instruction.
A 64-bit product is generated and the Y register receives the upper 32 bits of the product; register $g6 receives the lower 32 bits.
`0x4592be8*0x0e1e1000 = 0x3d631f67ee8000`
So:
Y contains `0x003d631f` and `%g6 contains 0x67ee8000`

9. The instruction addcc %o2, immed, %o4 adds an immediate operand (immed) to register %o2 and places the sum into register %o4. If the assembly instruction is to be translated into a single SparcV8 machine instruction, what is the largest value that can be used for the immediate operand?
The immediate operand must fit into the signed 13-bit immediate field within the machine instruction. Hence the maximum value 4095.

10. A pair of 64-bit integers reside in memory at addresses NUM1 and NUM2, respectively. Write down a sequence of SparcV8 instructions that would compute the 64-bit integer sum of the two integers and store the result back into memory at address NUM3. The three double words (NUM1, NUM2 and NUM3) are not adjacent in memory.

```
set     NUM1, %o2          put address of NUM1 into %o2
ldd     [%o2], %o4         load NUM1 into %o4 and %o5
set     NUM2, %o2          put address of NUM2 into %o2
ldd     [%o2], %o6         load NUM2 into %o6 and %o7
addcc   %o5, %o7, %o7      sum of low parts (& set condition codes)
addx    %o4, %o6, %o6      %o6 = sum of high parts plus carry bit
set     NUM3, %o2          put address of NUM3 into %o2
std     %o6, [%o2]         store %o6 and %o7 into NUM3
```