# 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
```

Module 13 Example Set 1

1. Shown below is an algorithm for determining the greatest common divisor (GCD) of two integers A and B each of which is greater than zero:

```
Set X = A
Set Y = B
While(X ≠ Y)
{
   if X>Y   set X=X-Y
  else if X<Y set  Y=Y-X
}
GCD = X
```

Show how a relatively small number of ARM conditional execution instructions can implement this algorithm.

Answer: Let R2 contain one number and R3 contain the other.

```
LOOP  CMP       R2,R3          ; compare number to set condition codes
      SUBGT     R2,R2,R3       ; if R2>R3,  R2 = R2 – R3
      SUBLT     R3,R3,R2       ; else if R3>R2,  R3 = R3 – R2
      BNE       LOOP           ; repeat loop until  R2 = R3
```

The CMP instruction sets the condition codes based on which register contains a larger number. SUBGT only executes if the condition codes set by CMP indicate that R2>R3. SUBLT only executes if the condition codes set by CMP indicate that R3>R2. BNE does not branch if the condition codes indicate that R3 = R2, in which case the branch back to LOOP is not taken and the GCD is contained in the two registers R3 and R2.

2. What are some of the advantages of the banked registers available on ARM processors compared to other RISC processors that do not employ banked registers?
The banked registers reduce the need to save and store registers to and from the stack, thus speeding up the processing of exceptions.

3. How does the process of identifying the proper exception handling routine differ on the ARM processor from that used on the MIPS processor?
On the MIPS processor all exceptions cause a transfer to the same exception vector address. Code that resides at that address must examine the cause register to determine the appropriate handling routine to call for the type of exception. On the ARM processor, each exception type is vectored to a different location in low memory. Each location contains an instruction that branches to the corresponding exception handling routine.

4. The ARM processor, unlike the MIPS processor, supports a PC-relative addressing mode for memory operands. What is a major disadvantage of this addressing mode?
One disadvantage is that the memory operands must be located in the same area as the program code and must be close enough to referenced using an 8-bit displacement. Using instead an addressing mode such as register indirect or base register mode would allow the operand to be located anywhere within the 32-bit address space.

5. What effect is produced by the following ARM instruction?

```
SUBS   PC,R14_irq,#4
```

When the result register is the PC, the S suffix causes the processor to copy the contents of SPSR_irq into CPSR. Also the instruction subtracts 4 from R14_irq and places the result into the PC causing control to be returned to the instruction that was about to execute when the interrupt occurred.

6. How does the process of enabling interrupts on the MIPS and ARM processors differ?
To enable interrupts, both processors have to be running in privileged mode.
On the MIPS, the status register (CP0 register 12) can be read and written using the instructions mfc0 and mtc0. Setting the LSB of CP0 $12 enables interrupts.
With the ARM processor running in priviledge mode, special Move instructions, MRS and MSR, can be used to transfer the contents of the processor status register to or from a CPU general purpose register.

```
MRS    R2,CPSR        ;  copies CPSR into R2

MSR    SPSR,R3        ; copies R3 into SPSR_mode
```

After status register contents have been loaded into a CPU register, logic instructions can be used to manipulate individual bits.

7. What is the purpose of the ARM instruction ADR?
This instruction loads into a register the address of operands that can be referenced using PC-relative addressing. That is, operands whose address can be generated adding or subtracting an 8-bit unsigned displacement (possibly shifted) to or from the PC. E.g. `ADR    R2,DATA1` places the PC-relative address of DATA1 into R2.

Module 13 Example Set 2

1. Register EBP contains the memory address of a sixty-four bit two's complement integer on an IA-32 processor system. The address in EBP corresponds to the low part of the 64-bit integer. Show a sequence of IA-32 instructions that subtract the 64-bit integer in memory from the 64-bit integer contained in the EDX,EAX register pair (EAX contains the low part of the subtrahend).

```
SUB     EAX,[EBP]           subtract the low parts
SBB     EDX,[EBP+4]         subtract high parts including any borrow
```

2. What is the effect produced by the following IA-32 instruction?

```
PADDB           MM2,MM4
```

This instruction interprets the contents of the MM2 and MM4 registers as a packed group of eight bytes and generates eight 8-bit sums by adding in parallel the corresponding bytes in the two registers.

3. What is the 80-bit extended precision floating point representation of the decimal number -2.75? Express the answer is hex.

The value is negative, so the sign bit = 1.
2.75 decimal = binary 10.11 = 1.011*$2^1$
So the fraction is 011 followed by sixty 0 bits
The characteristic = 1 + 16383 = 16384 = 0x4000
So the 80-bit representation is
1 100000000000000 1 011 followed by sixty 0 bits
or 0xC000B000000000000000

4. If the IA-32 floating point register ST(0) contains the floating point representation of the decimal number -2.75, what effect does the following instruction have? Show in hex, the contents of any memory locations that are changed.

```
FISTP     [EAX]
```

This instruction stores the integer part of -2.75 into memory as a 32-bit two's complement integer. The integer is stored at the memory address contained in the EAX register. The 32-bit pattern stored is 0xFFFFFFFE. The "P" suffix means that ST(0) will be popped from the floating point stack.

5. Assume that prior to executing the instruction shown below, EBP contains the number 0x4ABC4, ESI contains 0x20 and EAX contains 0xFFFFFFF0. What is the contents of these three registers after the instruction is executed?

```
LEA   EAX,[EPB + ESI*4 + 60]
```

This instruction overwrites the EAX register with the value  0x4ABC4 + 0x20*4 + 60 =  0x4ABC4 + 0x80 + 0x3C = 0x4AC80

6. Show, in hex, the final contents of any data registers that are modified by the following instructions:

```
MOV    EAX,-511
CDQ
```

The MOV instruction puts the two's complement representation of -511 into the EAX register. The CDQ instruction uses sign extension to convert the 32-bit signed integer in EAX into the equivalent 64-bit signed integer in the EDX,EAX register pair. That is, the 32-bit quadword in EAX is converted into the equivalent double quadword in EDX,EAX.
So  EDX = 0xFFFFFFFF  and EAX = 0xFFFFFE01