

Reading: Mano sec. 5-7 and 6-8, reread chapters 5, 6, and 8 as needed to complete their topics. Patterson and Hennessy: “Skim” Chapter 8 and begin to study section 8.4. Problems like 8.16 will be assigned after the April break.

Problem 1: This problem is are to write two assembly language codes for the two assignment statements which comprise the body of the following C subroutine<sup>1</sup>. The program should ignore overflow. (Omit everything involved with the call and return operations, even if you know about them). **EVERY assembly language line must have a comment that expresses its purpose and demonstrates that you understood why you wrote that line. Lines without comments will cause points to be deducted.**

```
int A; int B; int C; int D; int X;

void doit(void)
{
    X = (A + B)*(C + D);
    B = A*B + C*D;
}
```

Variables A,B,C,D and X are memory words. Refer to their addresses by A,B,C,D and X as assembly language symbols. For example, for the Mano basic problem, assume that your program file will contain the statements<sup>2</sup>:

```
A,  HEX  ---
B,  HEX  ---
C,  HEX  ---
D,  HEX  ---
X,  HEX  ---
```

- a. Show the assembly language code for the “Version M” Mano basic machine which provides an memory type integer multiply instruction named MUL<sup>3</sup> You can get started by referring to page 259 of Mano, but you must use the assembly language of chapters 5-6.
- b. Show the assembly language code for the MIPS machine, as described in Patterson and Hennessy.

It will be necessary to get the 32 bit address values symbolized by A, etc. into registers. To say get address A into say register \$s0, use the instructions lui \$s0, hi16(A) followed by addi \$s0, \$s0, lo16(B). The assembler operator hi16(A) applied to symbol A evaluates to the high 16 bits of the value of the symbol (which is the address). Similarly, lo16(A) evaluates to the low 16 bits. See page 147 of Patterson and Hennessy about loading a 32 bit constant. (The notation hi16() and lo16() are created for this problem. A real MIPS assembler has a “load address” pseudo-instruction that assembles to a sequence of two machine instructions.) Look up mult in the index and be sure to use the *move from lo mfl0* instruction to move the product to a general purpose register after the mult instruction. Assume ints are 32 bits long.

---

<sup>1</sup>All C code in this homework is also C++ code  
<sup>2</sup>Replace “---” with “0000” if your assembler doesn’t like “---”  
<sup>3</sup>Please ignore the fact that there are not enough opcodes for this.

Problem 2: The next problem is again to write two fully commented assembly language codes. This time, they are for the loop comprises the body of the following C subroutine. Use the same assumptions as those for the first problem. In particular, `psrc`, `pdest` and `temp` should be used (like A, B, C, D and X) as labels that will symbolize addresses of the 3 memory locations that will hold the values of these C language variables.

```
int *psrc;    //Pointer to an array of integers--Source for the copy
int *pdest;  //Pointer to another array of integers--Destination
int temp;    //Integer variable to hold last value copied

// copy an zero terminated array of integers, including the terminating zero
void intloop(void) {
    do {
        temp = *psrc;
        *pdest = temp;
        psrc++;
        pdest++;
    } while ( temp != 0 );
}
```

- a. Show the Mano basic computer assembly language code.
- b. Show the MIPS assembly language code.

Practice Problem A:

- a. Suppose a Mano basic computer is started with the first two memory locations holding 2000, 7010. What is the contents of the instruction register IR at the end of each clock step  $T_0$ ,  $T_1$ ,  $T_2$  and  $T_3$  of 7010's instruction cycle? Why doesn't IR=7010 at the end of  $T_0$ ?
- b. Instruction 7010 is a register transfer type instruction, so the field (0-11) containing 010<sub>Hex</sub> is **not** used as a memory address. However, in step  $T_2$ , 010 is transferred to the address register AR anyway. Explain. What would be required to avoid this transfer and would there be any advantage at all to avoiding it?
- c. Mano p. 168 problem 5-7.

Practice Problem B: Imagine 4 successive bytes of data in a computer memory. Write down as many different interpretations these 32 bits of data may have, and for each, give a concrete example of 32 bits and their interpretation.

Examples: Signed 2-s complement binary integer, unsigned binary, an array of 4 signed 2-s complement binary integers, an array of 4 ASCII characters, 2 Mano Basic instructions, one or more instructions of another instruction set architecture (you name some you know), some other numerical interpretations, first 32 bits of a TCP/IP network packet message (see page 653 of Patterson and Hennessy), some other kind of code such as Gray, ...

Write down 32 bit bit string whose high order bit is 1. Calculate and write in signed decimal its two interpretations: one interpretation as a signed 2-s complement binary numeral and the other as an unsigned binary numeral.

Create 3 BDDs: One to tell if a length 8 bit string represents 0 in 2-s complement binary, another to tell if it represents a non-zero positive number, and a third to tell if it is negative. Translate these BDDs to logic circuit diagrams.

Problem 3: Mano p. 169 problem 5-14. "Making changes to the basic computer."

Problem 4: Write a subroutine to read a specified number of characters into a specified area of memory. This operation should be **blocking**. That means the subroutine will not return until the operation is complete. For simplicity, store the characters unpacked, i.e., each 16 bit word has one character in its low 8 bits. The READN subroutine should use **busy waiting** (no interrupts!)

```

        BSA    READN    /Example of using READN
        HEX    MY_BUFFER /Pointer to input area
        HEX    n        /Number of characters to read
        ...
        ...
        /READN returns here
        ...
MY_BUFFER,
        ...
        /at least n words reserved here
        ...
READN,   HEX    ---    /Start of READN subroutine
        ...
        /YOU write the code here
        ...
        ISZ    READN
        ISZ    READN    /must increment by 2 before return!
        BUN    READN I  /READN returns
        ...
        /space for READN's variables

```

- a. Write an algorithm outline for READN in clear English.
- b. Write the Mano Basic Assembly language code, with ample comments.

Problem 5:

```

        ORG    0
ZERO,   HEX    0
        INP
        ION
        BUN    ZERO I
JUNK,   HEX    99
START,  ION
        ISZ    JUNK    /FGI goes ON near the beginning
        /of the clock step T2 of this ISZ instruction
        STA    JUNK
        HLT

```

Write every clock step and all the micro-operations that are done in that clock step when the Mano Basic computer is started with PC=START and the memory containing the above program. Make sure your answer indicates which instruction is being executed for each sequence of clock steps, and which clock steps are taken to handle the interrupt. **But before you begin**, note the comment on the ISZ instruction and **make the assumption that it specifies**. (The ISZ instruction will be executed only once, if the program is interpreted correctly.) What is M[JUNK] when the computer halts?