PROBLEMS

6-1. The following program is stored in the memory unit of the basic computer. Show the contents of the *AC*, *PC*, and *IR* (in hexadecimal), at the end, after each instruction is executed. All numbers listed below are in hexadecimal.

Location	Instruction
010	CLA
011	ADD 016
012	BUN 014
013	HLT
014	AND 017
015	BUN 013
016	C1A5
017	93C6

6-2. The following program is a list of instructions in hexadecimal code. The computer executes the instructions starting from address 100. What are the content of *AC* and the memory word at address 103 when the computer halts?

Location	Instruction
100	5103
101	7200
102	7001
103	0000
104	7800
105	7020
106	C103

6-3. List the assembly language program (of the equivalent binary instructions) generated by a compiler from the following Fortran program. Assume integer variables.

```
SUM = 0
SUM = SUM + A + B
DIF = DIF - C
SUM = SUM + DIF
```

6-4. Can the letter I be used as a symbolic address in the assembly language program defined for the basic computer? Justify the answer.

6-5. What happens during the first pass of the assembler (Fig. 6-1) if the line of code that has a pseudoinstruction ORG or END also has a label? Modify the flowchart to include an error message if this occurs.

6-6. A line of code in an assembly language program is as follows:

```
DEC -35
```

a. Show that four memory words are required to store the line of code and give their binary content.

b. Show that one memory word stores the binary translated code and give its binary content.

6-7. a. Obtain the address symbol table generated for the program of Table 6-13 during the first pass of the assembler.

b. List the translated program in hexadecimal.

6-8. The pseudoinstruction BSS N (block started by symbol) is sometimes employed to reserve *N* memory words for a group of operands. For example, the line of code

```
A, BSS 10
```

informs the assembler that a block of 10 (decimal) locations is to be left free, starting from location A. This is similar to the Fortran statement DIMENSION A(10). Modify the flowchart of Fig. 6-1 to process this pseudoinstruction.

6-9. Modify the flowchart of Fig. 6-2 to include an error message when a symbolic address is not defined by a label.

6-10. Show how the MRI and non-MRI tables can be stored in memory.

6-11. List the assembly language program (of the equivalent binary instructions) generated by a compiler for the following IF statement:

```
IF(A - B) 10, 20, 30
```

The program branches to statement 10 if $A - B < 0$; to statement 20 if $A - B = 0$; and to statement 30 if $A - B > 0$.

6-16. Write a program to multiply two unsigned positive numbers, each with 16 significant bits, to produce an unsigned double-precision product.

6-17. Write a program to multiply two signed numbers with negative numbers being initially in signed-2's complement representation. The product should be single-precision and signed-2's complement representation if negative.

6-18. Write a program to subtract two double-precision numbers.

6-19. Write a program that evaluates the logic exclusive-OR of two logic operands.

6-20. Write a program for the arithmetic shift-left operation. Branch to OVF if an overflow occurs.

6-21. Write a subroutine to subtract two numbers. In the calling program, the BSA instruction is followed by the subtrahend and minuend. The difference is returned to the main program in the third location following the BSA instruction.

6-22. Write a subroutine to complement each word in a block of data. In the calling program, the BSA instruction is followed by two parameters: the starting address of the block and the number of words in the block.

6-23. Write a subroutine to circulate E and AC four times to the right. If AC contains hexadecimal 079C and $E = 1$, what are the contents of AC and E after the subroutine is executed?

6-24. Write a program to accept input characters, pack two characters in one word and store them in consecutive locations in a memory buffer. The first address of the buffer is $(400)_{16}$. The size of the buffer is $(512)_{10}$ words. If the buffer overflows, the computer should halt.

6-25. Write a program to unpack two characters from location WRD and store them in bits 0 through 7 of locations CH1 and CH2. Bits 9 through 15 should contain zeros.

6-26. Obtain a flowchart for a program to check for a CR code (hexadecimal 0D) in a memory buffer. The buffer contains two characters per word. When the code for CR is encountered, the program transfers it to bits 0 through 7 of location LNE without disturbing bits 8 through 15.