1. Write a sequence of instructions for SIC to ALPHA equal to the product of BETA and GAMMA. Assume that ALPHA, BETA and GAMMA are defined as in Fig.1.3(a).

Assembly Code:

```
LDA    BETA
MUL    GAMMA
STA    ALPHA
:
:
ALPHA RESW 1
BETA  RESW 1
GAMMA RESW 1
```

2. Write a sequence of instructions for SIC/XE to set ALPHA equal to 4 * BETA – 9. Assume that ALPHA and BETA are defined as in Fig. 1.3(b). Use immediate addressing for the constants.

Assembly Code:

```
LDA    BETA
LDS     #4
MULR S,A
SUB    #9
STA    ALPHA
:
:
ALPHA RESW 1
```

3. Write SIC instructions to swap the values of ALPHA and BETA.

Assembly Code:

```
LDA     ALPHA
STA    GAMMA
LDA    BETA
STA    ALPHA
LDA    GAMMA
STA    BETA
:
:
ALPHA RESW 1
BETA  RESW 1
GAMMA RESW 1
```

4. Write a sequence of instructions for SIC to set ALPHA equal to the integer portion of BETA ÷ GAMMA. Assume that ALPHA and BETA are defined as in Fig.1.3(a).

Assembly Code:

```
LDA    BETA
DIV    GAMMA
STA    ALPHA
:
:
ALPHA RESW 1
BETA  RESW 1
GAMMA RESW 1
```

5. Write a sequence of instructions for SIC/XE to divide BETA by GAMMA, setting ALPHA to the integer portion of the quotient and DELTA to the remainder. Use register-to-register instructions to make the calculation as efficient as possible.

Assembly Code:

```
LDA    BETA
LDS    GAMMA
DIVR   S, A
STA    ALPHA
MULR S, A
LDS    BETA
SUBR A, S
STS    DELTA
:
:
ALPHA RESW 1
BETA  RESW 1
GAMMA RESW 1
DELTA RESW 1
```

6. Write a sequence of instructions for SIC/XE to divide BETA by GAMMA, setting ALPHA to the value of the quotient, rounded to the nearest integer. Use register-to-register instructions to make the calculation as efficient as possible.

Assembly Code:

```
LDF     BETA
DIVF    GAMMA
FIX
STA     ALPHA

:
:
ALPHA RESW 1
BETA   RESW 1
GAMMA RESW 1
```

7. Write a sequence of instructions for SIC/XE to clear a 20-byte string to all blanks.

Assembly Code:

```
LDX     ZERO
LOOP LDCH BLANK
STCH  STR1,X
TIX      TWENTY
JLT      LOOP
:
:
STR1   RESW 20
BLANK BYTE C ‘ ‘
ZERO WORD 0
TWENTY WORD 20
```

8. Write a sequence of instructions for SIC/XE to clear a 20-byte string to all blanks. Use immediate addressing and register-to-register instructions to make the process as efficient as possible.

Assembly Code:

```
LDT     #20
LDX     #0
LOOP LDCH #0
STCH  STR1,X
TIXR   T
JLT      LOOP
:
:
STR1   RESW 20
```

9. Suppose that ALPHA is an array of 100 words, as defined in Fig. 1.5(a). Write a sequence of instructions for SIC to set all 100 elements of the array to 0.

Assembly Code:

```
LDA    ZERO
STA    INDEX
LOOP  LDX INDEX
LDA    ZERO
STA    ALPHA, X
LDA    INDEX
ADD    THREE
STA    INDEX
COMP K300
TIX      TWENTY
JLT      LOOP
:
:
INDEX RESW 1
ALPHA RESW 100
:
ZERO  WORD  0
K300   WORD 100
THREE WORD 3
```

10. Suppose that ALPHA is an array of 100 words, as defined in Fig. 1.5(a). Write a sequence of instructions for SIC/XE to set all 100 elements of the array to 0. Use immediate addressing and register-to-register instructions to make the process as efficient as possible.

Assembly Code:

```
LDS     #3
LDT     #300
LDX     #0
LOOP  LDA  #0
STA     ALPHA, X
ADDR  S, X
COMPR X, T
JLT      LOOP
:
:
ALPHA RESW 100
```

11. Suppose that ALPHA is an array of 100 words. Write a sequence of instruction for SIC/XE to arrange the 100 words in ascending order and store result in an array BETA of 100 elements.

Assembly Code:

NOT YET SOLVED

12. Suppose that ALPHA and BETA are the two arrays of 100 words. Another array of GAMMA elements are obtained by multiplying the corresponding ALPHA element by 4 and adding the corresponding BETA elements.

Assembly Code:

```
LDS     #3
LDT     #300
LDX     #0
ADDLOOP LDA ALPHA, X
MUL     #4
ADD     BETA, X
STA     GAMMA, X
ADDR    S, X
COMPR X, T
JLT     ADDLOOP
:
:
ALPHA RESW 100
BETA  RESW 100
GAMMA RESW 100
```

13. Suppose that ALPHA is an array of 100 words. Write a sequence of instructions for SIC/XE to find the maximum element in the array and store results in MAX.

Assembly Code:

```
LDS     #3
LDT     #300
LDX     #0
CLOOP LDA ALPHA, X
COMP MAX
JLT     NOCH
STA     MAX
NOCH ADDR S, X
COMPR X, T
JLT     CLOOP
:
:
ALPHA RESW 100
MAX WORD -32768
```

14. Suppose that RECORD contains a 100-byte record, as in Fig. 1.7(a). Write a subroutine for SIC that will write this record on to device 05.

Assembly Code:

```
JSUB   WRREC
:
:
WRREC LDX ZERO
WLOOP TD OUTPUT
JEQ    WLOOP
LDCH   RECORD, X
WD     OUTPUT
TIX    LENGTH
JLT    WLOOP
RSUB
:
:
ZERO WORD 0
LENGTH WORD 1
OUTPUT BYTE X ‘05’
RECORD RESB 100
```

15. Suppose that RECORD contains a 100-byte record, as in Fig. 1.7(a). Write a subroutine for SIC that will write this record on to device 05.

Assembly Code:

```
JSUB   WRREC
:
:
WRREC LDX #0
LDT    #100
WLOOP TD OUTPUT
JEQ    WLOOP
LDCH   RECORD, X
WD     OUTPUT
TIXR   T
JLT    WLOOP
RSUB
:
:
OUTPUT BYTE X ‘05’
RECORD RESB 100
```

16. Write a subroutine for SIC that will read a record into a buffer, as in Fig.1.7(a). The record may be any length from 1 to 100 bytes. The end of record is marked with a “null” chara cter (ASCII code 00). The subroutine should place the length of the record read into a variable named LENGTH.

Assembly Code:

```
JSUB   RDREC
:
:
RDREC LDX ZERO
RLOOP TD INDEV
JEQ     RLOOP
RD      INDEV
COMP NULL
JEQ     EXIT
STCH  BUFFER, X
TIX      K100
JLT      RLOOP
EXIT    STX LENGTH
RSUB
:
:
ZERO  WORD  0
NULL   WORD 0
K100   WORD 1
INDEV BYTE X ‘F1’
LENGTH RESW 1
BUFFER RESB 100
```

17. Write a subroutine for SIC/XE that will read a record into a buffer, as in Fig.1.7(a). The record may be any length from 1 to 100 bytes. The end of record is marked with a “null” character (ASCII code 00). The subroutine should place the length of the record read into a variable named LENGTH. Use immediate addressing and register-to-register instructions to make the process as efficient as possible.

Assembly Code:

```
JSUB RDREC
:
:
RDREC LDX   #0
LDT     #100
LDS     #0
RLOOP TD INDEV
JEQ     RLOOP
RD      INDEV
COMPR A, S
JEQ     EXIT
STCH  BUFFER, X
TIXR T
JLT RLOOP
EXIR STX LENGTH
RSUB
:
:
INDEV BYTE X ‘F1’
LENGTH RESW 1
BUFFER RESB 100
```