**SYSTEM PROGRAMMING PYQs**

***Authored By: Arkajyoti Naskar***

1. **What is the utility of the relocation bit? (5)**

There are two methods for specifying relocation in object program 1. Modification Record 2. Relocation Bit. To overcome the disadvantage of modification record, relocation bit is used.

The relocation bit, or R bit, is a bit in the object code of a program that is used to indicate whether or not a particular instruction or data item can be relocated. A relocation bit is associated with each word of object code. Since all SIC instructions occupy one word, this means that there is one relocation bit for each possible instruction. The relocation bits are. gathered together into a bit mask following the length indicator in each Text record.

col 1: T

col 2-7: starting address

col 8-9: length (byte)

col 10-12: relocation bits

col 13-72: object code

**Text record format**

*If the R bit is set to 1*, it indicates that the instruction or data item can be relocated when the program is loaded into memory.

*If the R bit is set to 0*, it indicates that the instruction or data item cannot be relocated and must remain at its current address. If a Text record contains fewer than 12 words of object code, the bits corresponding to unused words are set to 0.

The utility of the R bit is to allow programs to be more flexible and adaptable, and to reduce the size of object code files.

1. **Differentiate between Rep and REPT directives with proper examples. (8)**

REP and REPT are two different directives that are used in assembly language programming. The REPT directive is used to repeat a block of code a specified number of times. A REPT does not have a name, so it cannot be called like a macro.

define macro count

value=0

REPT count

db value

value=value+1

endm

endm

Now this can be called several times.

list label byte

define 4

define 3

The REP directive is not a standalone directive in assembly language. Instead, it is commonly used as a prefix for certain to repeat a single instruction a specified number of times while a specific condition is met i.e. until CX=0.

MOV SI, source\_address

MOV DI, destination\_address

MOV CX, length of string

REP MOVSB

1. **How does DLL work on the Windows platform? (7)**

A dynamic-link library (DLL) is a file that contains code and data that can be used by multiple programs at the same time. DLLs are an important part of the Windows operating system, as they provide a way for programs to share common code and resources. By using a DLL, a program can be modularized into separate components.

When a program needs to use a DLL, it typically calls a function exported by the DLL. The function performs the desired operation and returns a value to the calling program. The DLL can also provide data to the calling program, such as constants and global variables.

To use a DLL, a program must first load it into memory. This is typically done using the LoadLibrary function provided by the operating system. Once the DLL is loaded, the program can call its exported functions using the GetProcAddress function. This function allows the program to obtain the memory address of the exported function, which it can then call system directly.

Overall, DLLs provide a way for programs to share code and resources, improving efficiency and reducing the amount of memory and storage used by the system.

1. **“T0010001E141033” is an example of a Text Record. Write the format of Text Record and explain the above example.**

A text record, or T record, is a type of record used in object code files in assembly language programming. It is used to store the instructions and data that are to be loaded into memory when the program is executed.

The general format of a text record is as follows -> T<address><length><data>. In this format, "T" indicates that this is a text record, <address> is the starting address of the data in the record,<length> is the length of the data in the record, and <data> is the actual data stored in the record.

For example, the text record "T0010001E141033" can be broken down as follows:

● "T" indicates that this is a text record

● "001000" is the starting address of the data in the record (3 byte hex)

● "1E" is the length of the data in the record (30 bytes)

● "141033" is the data stored in the record (the hexadecimal representation of the data).

In this example, the text record contains 14 bytes of data starting at the address 0x10001E. The data itself is represented in hexadecimal and can be decoded to determine the actual instructions or data stored in the record.

1. **What are the advantages and disadvantages of Overlay. (8)**

Overlay is needed when a process is larger than the amount of memory allocated to it. Keep in memory only those instructions and data that are needed at any given time, other instructions and data are loaded into the memory space.

**Advantages of Overlay:**

● Allow programs larger than physical memory.

● No special support is required from the OS.

● Implemented by user.

● Reduce memory requirements.

● Reduce Time Requirements.

● Allows programs to adapt dynamically to varying execution paths and user interactions.

**Disadvantages of Overlay:**

● Process Execution is slower due to extra I/O.

● Overlays are limited to microcomputers.

● Special relocation and linking algorithms are needed to construct Overlays.

● Overlay map must be specified by the programmers. Programmer must know memory requirements.

● Overlay module must be completely disjoint.

● Programming design of overlay structure is complex and not possible in all cases.

1. **Why is an absolute loader is called as a self-destroyed loader? (5)**

Absolute Loader is a type of loader in which relocated object files are created, loader accepts those files and places them at specified locations in memory. This is called absolute loader because no relocation information is needed. The starting address of every module is known to the programmer, this corresponding starting address is stored in the object file, then task of loader becomes very simple and that is to simply place the executable form of the machine instructions at the locations mentioned in

the object file. In this scheme the programmer or the assembler should have knowledge of memory management.

It does not perform linking and program relocation. The contents of memory locations for which there is no Text record are shown as xxxx. Each byte of assembled code is given using its Hex representation in character form.

The absolute loader is simple to implement in this scheme-

1. Allocation is done by either programmer or assembler.
2. Linking is done by the programmer or assembler.
3. Resolution is done by assembler.
4. Simply loading is done by the loader

As the name suggests, no relocation information is needed, if at all it is required then that task can be done by either a programmer or assembler

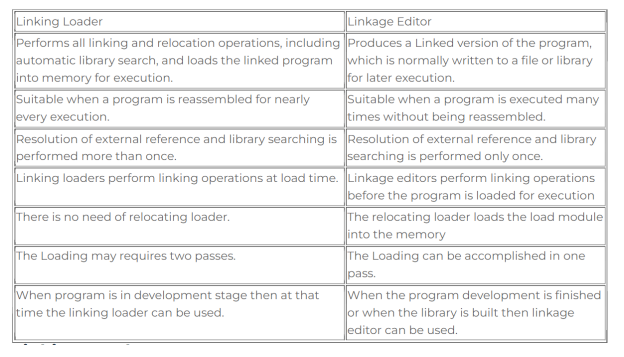
Absolute loader is known as self-destroyed loader as when the loader loads the object code in the fixed location, the code of the loader itself is overwritten, when the entire object program is loaded, the loader is destroyed.

**7. Compare Similarities and dissimilarities between linkage editor and loader-linker. (7)**

1. Linkage editor needs relocating loader to load the machine code into memory. But loader-linker loads the code into memory itself.

2. Both linkage editor and loader-linker are used to link object code after assembling.

3. Linkage editor produces a linked object file. Loader-linker produces no such file.



1. **What are different addressing modes of 8086?** (5)

***Immediate addressing mode:*** The addressing mode in which the data operand is a part of the instruction itself is known as immediate addressing mode.

Example: - MOV CX, 4929 H

ADD AX, 2387 H

MOV AL, FFH

***Register addressing mode***: It means that the register is the source of an operand for an instruction.

Example: - MOV CX, AX; copies the contents of the 16-bit AX register into; the 16-bit CX register),

ADD BX, AX

***Direct addressing mode***: The addressing mode in which the effective address of the memory location is written directly in the instruction.

Example: - MOV AX, [1592H]

MOV AL, [0300H]

***Register indirect addressing mode:*** This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI & SI.

Example: - MOV AX, [BX]; Suppose the register BX contains 4895H, then the contents ; 4895H are moved to AX

ADD CX, {BX}

***Based addressing mode: -*** In this addressing mode, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement. Example: - MOV DX, [BX+04]

ADD CL, [BX+08]

***Indexed addressing mode***: In this addressing mode, the operand’s offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements. Example: - MOV BX, [SI+16]

ADD AL, [DI+16]

***Based-index addressing mode***: In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.

Example: - ADD CX, [AX+SI]

MOV AX, [AX+DI]

***Based indexed with displacement mode***: In this addressing mode, the operands offset is computed by adding the base register contents. An Index registers contents and 8 or 16-bit displacement.

Example: - MOV AX, [BX+DI+08]

ADD CX, [BX+SI+16]

1. **Instructions of 8086 can execute parallel. – Justify it. Design a flowchart for one-pass assembler mentioning all data structure and steps clearly. (5+15=20)**

8086 employs parallel processing i.e., both BIU (Bus Interface Unit) and EU (Execution unit) work at the same time. This is unlike 8085 in which sequential fetch and execute operations take place.

The Execution unit gives instructions to BIU starting from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses, it performs operations over data through BIU.

BIU takes care of all data and address transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

[**Single pass assembler**](https://drive.google.com/file/d/1d4AbsugWZ_PRjpOTvFPnJO0QCrj-rpUl/view)

**9.** **What are advantages and disadvantages of holding symbolic operation codes in a separate symbol table? Write an Assembly language Program to find the number of vowels. (5+15=20)**

***Advantages :*** fast

***Disadvantages :*** Chance of collision if two label has same hash value.

**;8086 PROGRAM: COUNT NUMBER OF VOWELS IN GIVEN LINE OF A TEXT/SENTENCE**

.MODEL SMALL

.STACK 100H

.DATA

STRING DB 10,13,"The quick brown fox jumped over lazy sleeping dog$"

VOWEL DB ?

MSG1 DB 10,13,"Number of vowels are: $"

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV SI, OFFSET STRING

MOV BL, 00

BACK: MOV AL, [SI]

CMP AL,'$'

JZ FINAL

CMP AL,'A'

JZ COUNT

CMP AL,'E'

JZ COUNT

CMP AL,'I'

JZ COUNT

CMP AL,'O'

JZ COUNT

CMP AL,'U'

JZ COUNT

CMP AL,'a'

JZ COUNT

CMP AL,'e'

JZ COUNT

CMP AL,'i'

JZ COUNT

CMP AL,'o'

JZ COUNT

CMP AL,'u'

JZ COUNT

INC SI

JMP BACK

COUNT:

INC BL

INC SI

JMP BACK

FINAL:

MOV AH,2H

MOV DL, BL

INT 21H; print number of vowels

MOV AH, 4CH

INT 21H

MAIN ENDP

END

**10. What are the differences between line editor and screen editor? Explain the differences in user mode and kernel mode in device driver architecture. How UART device driver works in EXINU. (5+8+7=20)**

***Line editor:*** In a line editor the unit of editing consists of an entire line, i.e., to modify even a single character the user has to modify or re-enter the entire line. These are very primitive type of editors. For example, the editor edlin of DOS, ed and ex of UNIX etc.

***Screen Editor:*** Screen oriented editor allows the use of cursor keys and/or mouse to point to a position at which the next editing is to be carried out. All modern systems

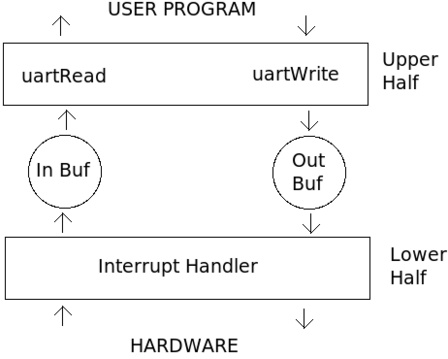
came up with one or more such Screen Oriented Editors such as vi, Emacs, etc.

***Differences between user mode and Kernel mode:***

|  |  |  |
| --- | --- | --- |
| **Property** | **User Mode** | **Kernel Mode** |
| Stability | Running the device driver in user mode can contribute to increased system stability.  Poorly written device drivers in user mode are less likely to crash the entire system because they operate within the constraints of user mode, which has restricted access to system resources. | Running the device driver in kernel mode poses a higher risk to system stability.  A poorly designed or malfunctioning kernel-mode device driver has the potential to crash the entire system, as it operates with elevated privileges and can directly access kernel memory. |
| Memory Access | Device drivers in user mode have limited access to system memory and cannot directly override kernel memory. | Device drivers in kernel mode have unrestricted access to the entire system memory, including kernel memory. This makes them more powerful but also more potentially risky. |
| Communi-  cation with Kernel | To interact with the kernel, device drivers in user mode register with subsystems to correspond to the result.  The user interface communicates with these device drivers using device files, which serve as a mechanism provided by the kernel for direct user-driver interaction. | Device drivers in kernel mode have direct access to the kernel and can communicate with it without the need for intermediaries like subsystems. |
| User-Driver Interface | Device files act as a means for the user interface to talk to device drivers in user mode. This provides a structured interface for user applications to interact with the underlying hardware. | Device drivers in kernel mode interact directly with the kernel, bypassing the need for user-level device files. |

**UART device driver in EXINU:-**

The UART driver is a char-oriented driver responsible for receiving and sending bytes of data asynchronously.

The UART driver is divided into two sections: **an upper half and a lower half.** The two halves communicate via semaphores and buffers. The lower half is interrupt driven and interacts with the physical hardware. The upper half of the driver interacts with user programs. It does not interact directly with the hardware nor does it spinlock while waiting for the hardware to be ready. The upper half waits on semaphores which are signalled by the lower half to indicate bytes of data or free space are available in the appropriate buffer.

Initialize defines the starting values for all members of the control block: statistical counts are zeroed, buffers are defined, and semaphores are allocated. Also, part of the initialization process is setting values in the control and status registers:

* Line control is set to 8 bits, no parity, 1 stop.
* Receiver FIFO full, transmit buffer empty, and receiver line status interrupts are enabled.
* Hardware FIFOs are enabled.
* Divisor Latch bits (high and low).

***Upper Half :*** Read is part of the upper half of the driver that fills a user supplied buffer with bytes from the input buffer filled by the lower half of the driver. If the input buffer is empty, read waits for the lower half to signal on the input semaphore and indicate bytes are available in the input buffer.

Write is part of the upper half of the driver and places bytes from a user supplied buffer into the output buffer read by the lower half of the driver. If there is no free space in the output buffer, write waits for the lower half to signal on the output semaphore and indicate free space is available in the output buffer.

***Lower Half:*** The interrupt handler is the lower half of the driver. Three different types of interrupts are handled by the lower half:

* Line or modem status.
* Receiver hardware FIFO trigger level.
* Transmitter hardware FIFO empty

**11. Summarize edit control features available for text editors. (10)**

***1. Moving the Cursor:*** The ability to navigate through the document using arrow keys, Home, End, Page Up, Page Down, and other navigation keys.

***2. Deleting:*** Deleting operations involve removing characters, lines, or blocks of text from the document. This can include deleting a single character, a word, a line, or a selected portion.

***3. Replacing:*** Replacing allows users to replace specific occurrences of text with new content. It often involves finding a target string and substituting it with another.

***4. Pasting:*** Pasting involves inserting content from the clipboard into the document. Users can copy text from one location and paste it elsewhere.

***5. Searching:*** Searching enables users to look for specific text within the document. This operation helps locate occurrences of a particular string or pattern.

***6. Searching and Replacing:*** Combining search and replace functionalities allows users to find specific text and replace it with another string throughout the document.

***7. Saving and Loading:*** Saving involves storing the current state of the document to a file, while loading retrieves a document from a file. These operations are crucial for preserving work and sharing documents.

***8. Miscellaneous (e.g., Quitting):*** Miscellaneous operations cover various actions, including quitting or closing the text editor. This category may also include other features like changing settings or preferences.

These operations collectively constitute the core functionalities of a text editor, providing users with the tools needed for effective text manipulation, navigation, and document management.

**12. What do you mean by Deferred Linking?**

In dynamic linking, the compiling and linking code into loadable programs are delayed until run time. This feature is used with system libraries such as languages subroutine libraries. This reduces memory requirements. Due to the delay in linking until runtime, it is also called Deferred Linking.

**13. What is the function of the following Loader option?**

CHANGE RDREC, READ

The CHANGE RDREC, READ option is a loader option that specifies the format of the input records to be loaded by the loader. It is used to indicate that the value of RDREC is to be changed to READ in the external symbol table, for example when the RDREC module is to be replaced by the READ module.

**14. Give an example of a one-byte instruction available in SIC/XE.**

FLOAT, it does the following

F ← [ A], i.e., value of Accumulator is converted to 48 bit and stored in the F register.

**15. Compare relative advantages and disadvantages for one pass and two pass assemblers.**

**Pros of one pass:**

1. It is faster than two pass as it generates executable in a single pass

2. Easy to write and implement due to its simple design as compared to two pass.

**Cons of one pass:**

1. Forward reference problem exists. In this problem operands are utilized before declaration; this assembler has no info about the operand address this is to be present in the final executable code.

2. All errors are not detected

**Pros of two pass:**

1. Solved forward reference problem by using first pass to generate a system table.

2. All errors are detected. Simple errors are solved in the first pass, complex errors are solved in the second pass.

**Cons of two pass:**

1. Slower due to the second pass.

2. Design and implementation are more sophisticated as compared to single pass.

**16. Immediate operands and literals are both ways of specifying an operand value in a source statement. When might each be preferable? (5)**

Immediate operands are specified directly in the machine instruction, while literal values are stored in memory and the address of this value is provided in the instruction.

**Immediate Operands:**

- Preferable for small values that can be represented in a limited number of bits.

- Efficient in terms of code size and execution speed.

- Suitable when memory space is a concern.

**Literals:**

- Preferable for larger or more complex values.

- Enhance code readability and maintainability.

- Suitable when emphasizing human-readable code is a priority.