BE CST 6th Semester Final Examination, 2006
Systems Programming (CST 605)

Answer any 5 questions

F.M. 100 Time: 3 hrs

1., a) What are the different steps taken in PASS 1 and 2 of an assembler. Discuss with simple examples. Also include in your answer the important data structure used and their details. [8]

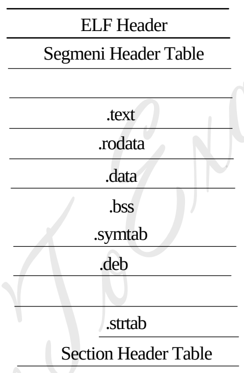
b) Explain with suitable examples the logic to be used to update the PEC on encountering the assembler directives ORG, EQU, DW and DB. [7]

c) An assembler supports an option by which it produces an alphabetical listing of all symbols used in a program. Comment on the suitability of different organisations for the symbol table using which the task may be done efficiently. [5]

2. a) Assume that you are in a programming environment where it is desirable to save all the registers (8085 CPU registers) upon entry to a subroutine and restore them upon exit. Write ENTRY and EXIT macros that could be called to do the job. [8]

b) Describe the five sections that are present in a typical object file. Also show that an absolutely simple object file format (DOS COM format) is possible which contains only the executable code. [5]

c) Adjoining figure shows typical headers and sections of an ELF object file. Elaborate the meaning of the headers and the sections. [7]



Typical ELF object file

3. a) Suppose that you are to write a "disassembler"- that is, a system program that takes an ordinary program as input and produces a listing of the source version of the program. What table and data structures would be required, and how would they be used? How many passes would be needed? What problem would arise in recreating the source program. [8]

b) Describe in detail the design of a macro-processor including the data structures required in its different passes. Assume that Macro call within a Macro as well as Macro definitions within another Macro are not required. [8]

c) What do you mean by a local symbol defined within a macro and how is that symbol expanded in the source file to avoid presence of multiple symbol in case of multiple calls to that macro? [4]

4. a) In UNIX environment a complier driver for any C program has 4 stages. What are these stages and what is done in each stage? [4]

b) "Object files in UNIX come in three different forms" - what are those forms and what are the characteristics of each of them? [4]

c) How linkers resolve multiply defined global symbols? [4]

d) Consider the following files addv.c and multv.c

```
/* addv.c */
void addvecdnt *x,   int  *y,   int  *z,   int n)
```

```
for  (i=0;  i < n;
```

```
/* multv.c */
void mulvec(int *x,   int *y,   int *z,   int n)

int  i;

for  (i=0;   i < n;   i
```

Now write the steps to create a static library and also show how this library can be used in a C program calling the function mulvec. Also show the steps to create the library as a shared object for dynamic linking. [4 + 4]

5. a) " The heart of a linker's (or loader's) action is relocation and code modification" - corroborate this statement with suitable example (s). [7]

b) EEF sections has types like progbits, nobits, symtab, dynsym etc. It also has attributes like AELOC, WRITE and EXECINSTR. Write the type and attributes for the following sections with explanation.

.text, .data, .rodata, .bss, .init and .fini [6]

c) Why library routines are used? Discuss in short the advantages and disadvantages of the unshared library, static shared library and dynamically shared librarv. Indexing; facility is added with the object ■files of the library for search efficiency. What would be the indexing for a tape library? [7]'

6. a) A program has the following 4 sections:

Name	Text	Data	bss
main	1502	302	65
ProcA	498	317	101
ProcB	1300	165	712
ProcC	256	1805	220

Considering all the data are in HEX, 4 byte word alignment and a page size of 0x1000 bytes show the run time layout assuming that the load point is 0x2000. [8]

b) Why does linker shuffle around segments to put segments of the same type next to each other? Would not be easier to leave them in the original order? [4]

c) Consider the following two source files; main.c and swap.c

```
/*   main.c   */   void
swapQ ; int buf[2]   =
{1,   2}

int main()
{
swapQ;
return 0;
```

```
/*   swap.c  */ extern
int  buf [] ;

int  *bufpO = &buf [0] ;
int  *bufpl;

void  swapQ int

temp ;

bufpi   =  febuf [1] ;
   temp  =   *bufpO;
• bufpO = *bufpl;
         "bufpl = temp;
```

List the global symbols and their types and other attributes in main.o and swap.o respectively. [8]

Write short notes on (Any four) [4 x 5]

(i) Weak Externals
(ii) Lazy binding
(Hi) Position Independent (.'ode
(iv) Expression evaluation in assembler; problem and solution
(v) Macro and Subroutine: similarity and difference
(vi) Functions to load and link shared object during execution