|  |  |
| --- | --- |
| 1A | Design suitable Data structures and implement Pass-I of assembler. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate codes file and symbol table).  **Input:**  START 100  A DS 3  L1 MOVER AREG, B  ADD AREG, C  MOVEM AREG, ='2'  MOVEM AREG, ='3'  D EQU A+1  LTORG  L2 PRINT D  MOVEM AREG, ='4'  MOVEM AREG, ='5'  ORIGIN L2+1  LTORG  B DC '19  C DC '17  END |
|  | Design suitable Data structures and implement Pass-II. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II.   |  |  | | --- | --- | | **Intermediate.txt**  (AD,01)(C,200)  (IS,04)(1)(L,1)  (IS,05)(1)(S,1)  (IS,04)(1)(S,1)  (IS,04)(3)(S,3)  (IS,01)(3)(L,2)  (IS,07)(6)(S,4)  (DL,01)(C,5)  (DL,01)(C,1)  (IS,02)(1)(L,3)  (IS,07)(1)(S,5)  (IS,00)  (AD,03)(S,2)+2  (IS,03)(3)(S,3)  (AD,03)(S,6)+1  (DL,02)(C,1)  (DL,02)(C,1)  (AD,02)  (DL,01)(C,1) | **Littab.txt**  5 206  1 207  1 213 | | **Symtab.txt**  A 1 211    LOOP 1 202    B 1 212    NEXT 1 208    BACK 1 202    LAST 1 210 | |
| 2A | Design suitable data structures and implement Pass-I of a two-pass macro-processor. The output of Pass-I (MNT, MDT and intermediate code file).  **Input:**  START  MACRO  INCR &ARG3 &ARG2  ADD AREG &ARG1  MOVER BREG &ARG1  MEND  MACRO  PVG &ARG2 &ARG1  SUB AREG &ARG2  MOVER CREG & ARG1  MEND  INCR  DECR  DATA2  END |
| 2B | Design suitable data structures and implement Pass-II of a two-pass macro-processor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.   |  |  | | --- | --- | | **Intermediate.txt**  M1 10,20,&b=CREG  M2 100,200,&u=&AREG,&v=&BREG | **Mnt.txt**  M1 2 2 1 1  M2 2 2 6 3 | | **Mdt.txt**  MOVE #3,#1  ADD #3,='1'  MOVER #3,#2  ADD #3,='5'  MEND  MOVER #3,#1  MOVER #4,#2  ADD #3,='15'  ADD #4,='10'  MEND | **Kpdt.txt**  a AREG  b -  u CREG  v DREG | |
|  |  |
| 4A | Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive) |
| 4B | Write a program to simulate CPU Scheduling Algorithms: FCFS, Priority (Non-Preemptive). |
| 4C | Write a program to simulate CPU Scheduling Algorithms: FCFS, Round Robin (Preemptive). |
| 4D | Write a program to simulate CPU Scheduling Algorithms: Priority (Non-Preemptive) and Round Robin (Preemptive). |
| 4E | Write a program to simulate CPU Scheduling Algorithms: SJF (Non Preemptive) and Round Robin (Preemptive) |
| 4F | Write a program to simulate CPU Scheduling Algorithms: Priority (Preemptive) and Round Robin (Preemptive). |
| 4G | Write a program to simulate CPU Scheduling Algorithms: SJF Non Preemptive) and Priority (Preemptive). |
| 4H | Write a program to simulate CPU Scheduling Algorithms: SJF (Preemptive) and Round Robin (Preemptive) |
| 5A | Write a program to simulate Memory placement strategies – first fit , next fit. |
| 5B | Write a program to simulate Memory placement strategies – first fit ,best fit. |
| 5C | Write a program to simulate Memory placement strategies – next fit and worst fit. |
| 5D | Write a program to simulate Memory placement strategies – best fit and worst fit. |
| 5E | Write a program to simulate Memory placement strategies – best fit, next fit. |
| 5F | Write a program to simulate Memory placement strategies – best fit, next fit. |
| 5G | Write a program to simulate Memory placement strategies – first fit , Worst fit. |
| 6A | Write a program to simulate Page replacement algorithm: FIFO, LRU |
| 6B | Write a program to simulate Page replacement algorithm: FIFO, Optimal |
| 6C | Write a program to simulate Page replacement algorithm: LRU, Optimal |
| 6D | Write a program to simulate Page replacement algorithm: LRU, Optimal |