CSL 331 SYSTEM SOFTWARE AND MICROPROCESSORS LAB

LABORATORY RECORD

SUBMITTED BY

MADHURYA R NAIR

NSS20CS029

ROLL NO: 26

to

the APJ Abdul Kalam Technological University

Fifth Semester B. Tech Degree Lab Examination (2019 Scheme)

Department of Computer Science and Engineering

N.S.S. College of Engineering, Palakkad

December 2022

SOURCE CODE:-

```
//FCFS
#include<stdio.h>
struct Process
{
	int pid;
	int at, bt, ct, wt, tt;
};
typedef struct Process pr;
void main()
{
	int n, i, j, t;
	float sumwt, sumtt;
	sumwt=0;
	sumtt=0;
	printf("\nEnter the number of processes: ");
	scanf("%d", &n);
	pr p[n], temp;
	printf("\nEnter the process id, arrival time and burst time of each process:\n");
	for (i=0 ; i<n ; i++)
		scanf("%d %d %d", &p[i].pid, &p[i].at, &p[i].bt);
	for (i=0 ; i<n-1 ; i++)
	{
		for (j=i+1 ; j<n ; j++)
		{
			if (p[i].at > p[j].at)
			{
				temp = p[i];
				p[i] = p[j];
				p[j] = temp;
			}
		}
```

```
        }
        t = p[0].at;
        for (i=0 ; i<n ; i++)
        {
                p[i].wt = t - p[i].at;
                t = t + p[i].bt;
                p[i].ct = t;
                p[i].tt = p[i].ct - p[i].at;
                sumwt += p[i].wt;
                sumtt += p[i].tt;
        }
        for (i=0 ; i<n-1 ; i++)
        {
                for (j=i+1 ; j<n ; j++)
                {
                        if (p[i].pid > p[j].pid)
                        {
                                temp = p[i];
                                p[i] = p[j];
                                p[j] = temp;
                        }
                }
        }
        printf("\nProcess ID   Arrival Time   Burst Time   Completion Time   Waiting Time
                                                                        Turnaround Time\n");
        for (i=0 ; i<n ; i++)
                printf("%5d%15d%14d%15d%15d%15d\n", p[i].pid, p[i].at, p[i].bt, p[i].ct, p[i].wt,
                                                                                p[i].tt);
        printf("\nAverage Waiting Time = %.3f", (sumwt/n));
        printf("\nAverage Turnaroung Time = %.3f\n", (sumtt/n));
}
```

OUTPUT:-

```
madhurya@madhurya:~/oslab$ gcc fcfs.c
madhurya@madhurya:~/oslab$ ./a.out

Enter the number of processes: 5

Enter the process id, arrival time and burst time of each process:
1 2 8
2 0 5
3 3 8
4 1 3
5 4 6

Process ID   Arrival Time   Burst Time   Completion Time   Waiting Time   Turnaround Time
    1             2             8             16              6              14
    2             0             5              5              0               5
    3             3             8             24             13              21
    4             1             3              8              4               7
    5             4             6             30             20              26

Average Waiting Time = 8.600
Average Turnaroung Time = 14.600
```

SOURCE CODE:-

```
//SJF
#include<stdio.h>
struct Process
{
	int pid;
	int at, bt, ct, wt, tt;
};
typedef struct Process pr;
void main()
{
	int n, i, j, t, b, pos;
	float sumwt, sumtt;
	sumwt=0;
	sumtt=0;
	printf("\nEnter the number of processes: ");
	scanf("%d", &n);
	pr p[n], temp;
	printf("\nEnter the process id, arrival time and burst time of each process:\n");
	for (i=0 ; i<n ; i++)
		scanf("%d %d %d", &p[i].pid, &p[i].at, &p[i].bt);
	for (i=0 ; i<n-1 ; i++)
	{
		for (j=i+1 ; j<n ; j++)
		{
			if (p[i].at > p[j].at)
			{
				temp = p[i];
				p[i] = p[j];
				p[j] = temp;
			}
		}
```

```
}
for (i=0 ; i<n ; i++)
{
        if (i != 0)
                t = p[i-1].ct;
        else
                t = p[i].at;
        b = p[i].bt;
        for (j=i ; j<n ; j++)
        {
                if ((p[j].at <= t) && (p[j].bt <= b))
                {
                        pos = j;
                        b = p[j].bt;
                }
        }
        p[pos].ct = t + p[pos].bt;
        p[pos].tt = p[pos].ct - p[pos].at;
        p[pos].wt = p[pos].tt - p[pos].bt;
        sumwt += p[pos].wt;
        sumtt += p[pos].tt;
        temp = p[pos];
        p[pos] = p[i];
        p[i] = temp;
}
for (i=0 ; i<n-1 ; i++)
{
        for (j=i+1 ; j<n ; j++)
        {
                if (p[i].pid > p[j].pid)
                {
                        temp = p[i];
```

```
                        p[i] = p[j];
                        p[j] = temp;
                    }
                }
            }
            printf("\nProcess ID   Arrival Time   Burst Time   Completion Time   Waiting Time
                                                                    Turnaround Time\n");
            for (i=0 ; i<n ; i++)
                printf("%5d%15d%14d%15d%15d%15d\n", p[i].pid, p[i].at, p[i].bt, p[i].ct, p[i].wt,
                                                                                    p[i].tt);
            printf("\nAverage Waiting Time = %.3f", (sumwt/n));
            printf("\nAverage Turnaroung Time = %.3f\n", (sumtt/n));
}
```

OUTPUT:-

```
madhurya@madhurya:~/oslab$ gcc sjf.c
madhurya@madhurya:~/oslab$ ./a.out

Enter the number of processes: 5

Enter the process id, arrival time and burst time of each process:
2 5 6
1 2 2
4 0 7
3 0 4
5 7 4

Process ID   Arrival Time   Burst Time   Completion Time   Waiting Time   Turnaround Time
    1             2             2             6              2              4
    2             5             6            12              1              7
    3             0             4             4              0              4
    4             0             7            23             16             23
    5             7             4            16              5              9

Average Waiting Time = 4.800
Average Turnaroung Time = 9.400
```

SOURCE CODE:-

```
//Priority
#include<stdio.h>
struct Process
{
	int pid;
	int at, bt, ct, wt, tt, pri;
};
typedef struct Process pr;
void main()
{
	int n, i, j, t, pos, prio;
	float sumwt, sumtt;
	sumwt=0;
	sumtt=0;
	printf("\nEnter the number of processes: ");
	scanf("%d", &n);
	pr p[n], temp;
	printf("\nEnter the process id, arrival time, burst time and priority of each process:\n");
	for (i=0 ; i<n ; i++)
		scanf("%d %d %d %d", &p[i].pid, &p[i].at, &p[i].bt, &p[i].pri);
	for (i=0 ; i<n-1 ; i++)
	{
		for (j=i+1 ; j<n ; j++)
		{
			if (p[i].at > p[j].at)
			{
				temp = p[i];
				p[i] = p[j];
				p[j] = temp;
			}
		}
```

```
}
for (i=0 ; i<n ; i++)
{
        if (i != 0)
                t = p[i-1].ct;
        else
                t = p[i].at;
        prio = p[i].pri;
        for (j=i ; j<n ; j++)
        {
                if ((p[j].at <= t) && (p[j].pri == prio))
                {
                        if (p[i].bt > p[j].bt)
                                pos = j;
                        else
                                pos = i;
                        prio = p[j].pri;
                }
                else if ((p[j].at <= t) && (p[j].pri < prio))
                {
                        pos = j;
                        prio = p[j].pri;
                }
        }
        p[pos].ct = t + p[pos].bt;
        p[pos].tt = p[pos].ct - p[pos].at;
        p[pos].wt = p[pos].tt - p[pos].bt;
        sumwt += p[pos].wt;
        sumtt += p[pos].tt;
        temp = p[pos];
        p[pos] = p[i];
        p[i] = temp;
```

```
        }
        for (i=0 ; i<n-1 ; i++)
        {
                for (j=i+1 ; j<n ; j++)
                {
                        if (p[i].pid > p[j].pid)
                        {
                                temp = p[i];
                                p[i] = p[j];
                                p[j] = temp;
                        }
                }
        }
        printf("\nProcess ID   Arrival Time   Burst Time   Completion Time   Waiting Time
                                                        Turnaround Time   Priority\n");
        for (i=0 ; i<n ; i++)
                printf("%5d%15d%14d%15d%15d%15d%14d\n", p[i].pid, p[i].at, p[i].bt, p[i].ct,
                                                                p[i].wt, p[i].tt, p[i].pri);
        printf("\nAverage Waiting Time = %.3f", (sumwt/n));
        printf("\nAverage Turnaroung Time = %.3f\n", (sumtt/n));
}
```

OUTPUT:-

```
madhurya@madhurya:~/oslab$ gcc prio.c
madhurya@madhurya:~/oslab$ ./a.out

Enter the number of processes: 5

Enter the process id, arrival time, burst time and priority of each process:
2 1 3 2
1 0 5 1
4 3 6 3
3 2 8 1
5 0 6 3

Process ID   Arrival Time   Burst Time   Completion Time   Waiting Time   Turnaround Time   Priority
    1             0             5             5              0              5              1
    2             1             3            16             12             15              2
    3             2             8            13              3             11              1
    4             3             6            22             13             19              3
    5             0             6            28             22             28              3

Average Waiting Time = 10.000
Average Turnaroung Time = 15.600
```

SOURCE CODE:-

```
//Banker's Algorithm
#include <stdio.h>
void main()
{
	int n, m, i, j, k, flag, safe=1, pos=0;
	printf("\nEnter the number of processes: ");
	scanf("%d", &n);
	printf("\nEnter the number of resource types: ");
	scanf("%d", &m);
	int maxr[m], max[n][m], alloc[n][m],need[n][m], avail[m], finish[n], safeseq[n];
	printf("\nEnter the instances of each resource:\n");
	for (i=0 ; i<m ; i++)
	{
		printf("Instances of resource R%d:", i);
		scanf("%d", &maxr[i]);
		avail[i]=maxr[i];
	}
	printf("\nEnter the maximum required resources of each process:\n");
	for (i=0 ; i<n ; i++)
	{
		printf("For P%d:\n",i);
		for (j=0 ; j<m ; j++)
		{
			printf("R%d:", j);
			scanf("%d", &max[i][j]);
		}
	}
	printf("\nEnter the allocation of each process:\n");
	for (i=0 ; i<n ; i++)
	{
		printf("For P%d:\n", i);
```

```
        for (j=0 ; j<m ; j++)
        {
                printf("R%d:", j);
                scanf("%d", &alloc[i][j]);
        }
}
printf("\nMaximum Matrix:\n");
for (i=0 ; i<n ; i++)
{
        for (j=0 ; j<m ; j++)
        {
                printf("%d", max[i][j]);
                printf("\t");
        }
        printf("\n");
}
printf("\nAllocation Matrix:\n");
for (i=0 ; i<n ; i++)
{
        for (j=0 ; j<m ; j++)
        {
                printf("%d", alloc[i][j]);
                printf("\t");
        }
        printf("\n");
}
printf("\nNeed Matrix:\n");
for (i=0 ; i<n ; i++)
{
        for (j=0 ; j<m ; j++)
        {
                need[i][j]=max[i][j]-alloc[i][j];
```

```
                avail[j]-=alloc[i][j];
                printf("%d", need[i][j]);
                printf("\t");
        }
        printf("\n");
}
for (i=0 ; i<n ; i++)
        finish[i] = 0;
for (i=0 ; i<n ; i++)
{
        for (j=0 ; j<n ; j++)
        {
                if (finish[j] == 0)
                {
                        flag = 0;
                        for (k=0 ; k<m ; k++)
                        {
                                if (need[j][k] > avail[k])
                                {
                                        flag = 1;
                                        break;
                                }
                        }
                        if (flag == 0)
                        {
                                safeseq[pos]=j;
                                pos++;
                                for (k=0 ; k<m ; k++)
                                        avail[k] += alloc[j][k];
                                finish[j] = 1;
                        }
                }
```

```
            }
        }
        for (i=0 ; i<n ; i++)
        {
            if (finish[i]==0)
            {
                safe=0;
                break;
            }
        }
        if(safe==0)
            printf("\nDeadlock occurred, System is in unsafe state\n");
        else
        {
            printf("\nSystem is in safe state\n");
            printf("The Safe Sequence is,\n");
            for (i=0 ; i<n ; i++)
                printf("P%d\t", safeseq[i]);
            printf("\n");
        }
}
```

OUTPUT:-

```
madhurya@madhurya:~/oslab$ gcc ba.c
madhurya@madhurya:~/oslab$ ./a.out

Enter the number of processes: 5

Enter the number of resource types: 4

Enter the instances of each resource:
Instances of resource R0:6
Instances of resource R1:7
Instances of resource R2:12
Instances of resource R3:12

Enter the maximum required resources of each process:
For P0:
R0:0
R1:0
R2:1
R3:2
For P1:
R0:2
R1:7
R2:5
R3:0
For P2:
R0:6
R1:6
R2:5
R3:6
For P3:
R0:4
R1:3
R2:5
R3:6
For P4:
R0:0
R1:6
R2:5
R3:2
```

```
Enter the allocation of each process:
For P0:
R0:0
R1:0
R2:1
R3:2
For P1:
R0:2
R1:0
R2:0
R3:0
For P2:
R0:0
R1:0
R2:3
R3:4
For P3:
R0:2
R1:3
R2:5
R3:4
For P4:
R0:0
R1:3
R2:3
R3:2

Maximum Matrix:
0       0       1       2
2       7       5       0
6       6       5       6
4       3       5       6
0       6       5       2

Allocation Matrix:
0       0       1       2
2       0       0       0
0       0       3       4
2       3       5       4
0       3       3       2

Need Matrix:
0       0       0       0
0       7       5       0
6       6       2       2
2       0       0       2
0       3       2       0

System is in safe state
The Safe Sequence is,
P0      P3      P4      P1      P2
```

SOURCE CODE:-

```
//Single Level Directory
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct directory
{
	char dname[10], fname[10][10];
	int fc;
}dir;
void list()
{
	int i;
	printf("\nThe files in the directory %s are: ", dir.dname);
	for (i=0 ; i<dir.fc ; i++)
		printf("%s\t", dir.fname[i]);
}
void main()
{
	int i, ch;
	char f[10];
	dir.fc=0;
	printf("\nEnter the directory name: ");
	scanf("%s", &dir.dname);
	printf("\nMAIN MENU:");
	printf("\n1: Create a file");
	printf("\n2: Delete a file");
	printf("\n3: Search a file");
	printf("\n4: List the files");
	printf("\n5: Exit");
	while(1)
	{
```

```
printf("\nEnter your choice: ");
scanf("%d", &ch);
switch(ch)
{
        case 1:
                y:
                printf("\nEnter the file to be created: ");
                scanf(" %s", &f);
                for (i=0 ; i<dir.fc ; i++)
                {
                        if (strcmp(f, dir.fname[i]) == 0)
                        {
                                printf("\nFile name already exists");
                                goto y;
                        }
                }
                strcpy(dir.fname[dir.fc], f);
                dir.fc++;
                printf("\nFile creation successful");
                list();
                break;
        case 2:
                printf("\nEnter the file to be deleted: ");
                scanf(" %s", &f);
                for (i=0 ; i<dir.fc ; i++)
                {
                        if (strcmp(f, dir.fname[i]) == 0)
                        {
                                printf("\nFile deletion successful");
                                strcpy(dir.fname[i], dir.fname[dir.fc-1]);
                                break;
                        }
```

```
                }
                if (i==dir.fc)
                        printf("\nFile %s not found", f);
                else
                {
                        dir.fc--;
                        list();
                }
                break;
        case 3:
                printf("\nEnter the file to be searched: ");
                scanf(" %s", &f);
                for (i=0 ; i<dir.fc ; i++)
                {
                        if (strcmp(f, dir.fname[i]) == 0)
                        {
                                printf("File %s found\n", f);
                                break;
                        }
                }
                if (i==dir.fc)
                        printf("File %s not found\n", f);
                break;
        case 4:
                list();
                break;
        case 5:
                printf("Exiting the menu\n");
                exit(0);
        default:
                printf("\nPlease enter a valid choice");
}
```

```
        }

}
```

OUTPUT:-

```
madhurya@madhurya:~$ ./a.out

Enter the directory name: class

MAIN MENU:
1: Create a file
2: Delete a file
3: Search a file
4: List the files
5: Exit
Enter your choice: 1

Enter the file to be created: name

File creation successful
The files in the directory class are: name
Enter your choice: 1

Enter the file to be created: section

File creation successful
The files in the directory class are: name      section
Enter your choice: 2

Enter the file to be deleted: name

File deletion successful
The files in the directory class are: section
Enter your choice: 3
```

```
Enter the file to be searched: name
File name not found

Enter your choice: 4

The files in the directory class are: section
Enter your choice: 5
Exiting the menu
```

SOURCE CODE:-

```
//Two Level Directory
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct directory
{
	char dname[10], fname[10][10];
	int fc;
}dir[100];
int dc=0;
void list()
{
	int i, j;
	printf("\n\nThe directories in the system are,");
	for (i=0 ; i<dc ; i++)
	{
		printf("\nDirectory Name: %s\n", dir[i].dname);
		if (dir[i].fc != 0)
		{
			printf("The files in the directory %s are: ", dir[i].dname);
			for (j=0 ; j<dir[i].fc ; j++)
				printf("%s\t", dir[i].fname[j]);
			printf("\n");
		}
		else
			printf("No files created yet\n");
	}
}
void main()
{
	int i, j, ch, flag;
```

```
char d[10], f[10];
for (i=0 ; i<100 ; i++)
        dir[i].fc=0;
printf("\nMAIN MENU:");
printf("\n1: Create a directory");
printf("\n2: Create a file");
printf("\n3: Delete a file");
printf("\n4: Search a file");
printf("\n5: List the files");
printf("\n6: Exit");
while(1)
{
        printf("\nEnter your choice: ");
        scanf("%d", &ch);
        switch(ch)
        {
                case 1:
                        x:
                        printf("\nEnter the directory name: ");
                        scanf(" %s", &d);
                        for (i=0 ; i<dc ; i++)
                        {
                                if (strcmp(d, dir[i].dname) == 0)
                                {
                                        printf("Directory name already exists");
                                        goto x;
                                }
                        }
                        strcpy(dir[dc].dname, d);
                        dc++;
                        printf("Directory creation successful");
                        list();
```

```
                    break;
        case 2:
                    flag=0;
                    printf("\nEnter the directory name: ");
                    scanf(" %s", &d);
                    for (i=0 ; i<dc ; i++)
                    {
                        if (strcmp(d, dir[i].dname) == 0)
                        {
                            flag=1;
                            y:
                            printf("\nEnter the file to be created: ");
                            scanf(" %s", &f);
                            for (j=0 ; j<dir[i].fc ; j++)
                            {
                                if (strcmp(f, dir[i].fname[j]) == 0)
                                {
                                    printf("File name already
                                    exists\n");
                                    goto y;
                                }
                            }
                            strcpy(dir[i].fname[dir[i].fc], f);
                            dir[i].fc++;
                            printf("File creation successful");
                            list();
                            break;
                        }
                    }
                    if (flag==0)
                        printf("Directory %s not found\n", d);
```

```
                    break;
            case 3:
                    flag=0;
                    printf("\nEnter the directory name: ");
                    scanf(" %s", &d);
                    for (i=0 ; i<dc ; i++)
                    {
                            if (strcmp(d, dir[i].dname) == 0)
                            {
                                    flag=1;
                                    printf("\nEnter the file to be deleted: ");
                                    scanf(" %s", &f);
                                    for (j=0 ; j<dir[i].fc ; j++)
                                    {
                                            if (strcmp(f, dir[i].fname[j]) == 0)
                                            {
                                                    printf("File deletion
                                                    successful");
                                                    strcpy(dir[i].fname[j],
                                                    dir[i].fname[dir[i].fc-1]);
                                                    break;
                                            }
                                    }
                                    if (j==dir[i].fc)
                                            printf("File %s not found\n", f);
                                    else
                                    {
                                            dir[i].fc--;
                                            list();
                                    }
                            }
                    }
```

```
                if (flag==0)
                        printf("Directory %s not found\n", d);
                break;
        case 4:
                flag=0;
                printf("\nEnter the directory name: ");
                scanf(" %s", &d);
                for (i=0 ; i<dc ; i++)
                {
                        if (strcmp(d, dir[i].dname) == 0)
                        {
                                flag=1;
                                printf("\nEnter the file to be searched: ");
                                scanf(" %s", &f);
                                for (j=0 ; j<dir[i].fc ; j++)
                                {
                                        if (strcmp(f, dir[i].fname[j]) == 0)
                                        {
                                                printf("File %s found\n", f);
                                                break;
                                        }
                                }
                                if (j==dir[i].fc)
                                        printf("File %s not found\n", f);
                        }
                }
                if (flag==0)
                        printf("Directory %s not found\n", d);
                break;
        case 5:
                list();
                break;
```

```
                case 6:
                        printf("Exiting the menu\n");
                        exit(0);
                default:
                        printf("\nPlease enter a valid choice");
            }
        }
}
```

OUTPUT:-

```
madhurya@madhurya:~$ ./a.out

MAIN MENU:
1: Create a directory
2: Create a file
3: Delete a file
4: Search a file
5: List the files
6: Exit
Enter your choice: 1

Enter the directory name: class
Directory creation successful

The directories in the system are,
Directory Name: class
No files created yet

Enter your choice: 1

Enter the directory name: college
Directory creation successful

The directories in the system are,
Directory Name: class
No files created yet
```

```
Directory Name: college
No files created yet

Enter your choice: 2

Enter the directory name: class

Enter the file to be created: name
File creation successful

The directories in the system are,
Directory Name: class
The files in the directory class are: name

Directory Name: college
No files created yet

Enter your choice: 2

Enter the directory name: college

Enter the file to be created: address
File creation successful

The directories in the system are,
Directory Name: class
The files in the directory class are: name

Directory Name: college
The files in the directory college are: address
```

```
Enter your choice: 3

Enter the directory name: college

Enter the file to be deleted: address
File deletion successful

The directories in the system are,
Directory Name: class
The files in the directory class are: name

Directory Name: college
No files created yet

Enter your choice: 4

Enter the directory name: class

Enter the file to be searched: name
File name found

Enter your choice: 5


The directories in the system are,
Directory Name: class
The files in the directory class are: name

Directory Name: college
No files created yet

Enter your choice: 6
Exiting the menu
```

SOURCE CODE:-

```
//Sequential File Allocation
#include<stdio.h>
#include<stdlib.h>
void main()
{
	int n, i, j, b[20], sb[20], mem[100], ch, count=0;
	for (i=0 ; i<100 ; i++)
		mem[i]=0;
	for (i=0 ; i<20 ; i++)
	{
		b[i]=0;
		sb[i]=0;
	}
	do
	{
		count++;
		printf("\nEnter the number of blocks occupied by file %d: ", count);
		scanf("%d", &b[count]);
		x:
		printf("\nEnter the starting block of file %d: ", count);
		scanf("%d", &sb[count]);
		for (j=sb[count] ; j<(sb[count]+b[count]) ; j++)
		{
			if (mem[j] == 0)
			{
				mem[j]=j;
				printf("Block %d has been allocated for file %d\n", j, count);
			}
			else
			{
				printf("\nBlock %d is already allocated", j);
```

```
                        printf("\nTry again!!\n");
                        for (int k=sb[count] ; k<j ; k++)
                                mem[k]=0;
                        goto x;
                }
        }
        printf("\nFile Allocation Successful\n");
        printf("\nPress 1 to continue allocation, Press 0 to stop allocation: ");
        scanf("%d", &ch);
    }
    while (ch==1);
    printf("\nFile\tStart\tBlock Length\n");
    for (j=1 ; j<=count ; j++)
        printf("%2d%9d%10d\n", j, sb[j], b[j]);
}
```

OUTPUT:-

```
Enter the number of blocks occupied by file 1: 2

Enter the starting block of file 1: 3
Block 3 has been allocated for file 1
Block 4 has been allocated for file 1

File Allocation Successful

Press 1 to continue allocation, Press 0 to stop allocation: 1

Enter the number of blocks occupied by file 2: 6

Enter the starting block of file 2: 4

Block 4 is already allocated
Try again!!

Enter the starting block of file 2: 5
Block 5 has been allocated for file 2
Block 6 has been allocated for file 2
Block 7 has been allocated for file 2
Block 8 has been allocated for file 2
Block 9 has been allocated for file 2
Block 10 has been allocated for file 2

File Allocation Successful

Press 1 to continue allocation, Press 0 to stop allocation: 0

File    Start   Block Length
 1        3         2
 2        5         6
```

SOURCE CODE:-

```
//Linked File Allocation
#include<stdio.h>
#include<stdlib.h>
void main()
{
	int i, j, n, mem[100], alloc[20], sb[20], b[20], count=0, ch, k, success;
	for (i=0 ; i<100 ; i++)
		mem[i]=0;
	for (i=0 ; i<20 ; i++)
	{
		alloc[i]=0;
		sb[i]=0;
		b[i]=0;
	}
	printf("\nEnter the number of blocks already allocated: ");
	scanf("%d", &n);
	printf("\nEnter the already allocated blocks:\n");
	for (i=0 ; i<n ; i++)
	{
		scanf("%d", &alloc[i]);
		mem[alloc[i]]=alloc[i];
	}
	do
	{
		count++;
		success=0;
		printf("\nEnter the number of blocks occupied by file %d: ", count);
		scanf("%d", &b[count]);
		printf("\nEnter the starting block of file %d: ", count);
		scanf("%d", &sb[count]);
		k=sb[count];
```

```
            while (success != b[count])
            {
                if (mem[k] == 0)
                {
                    mem[k]=k;
                    printf("Block %d has been allocated for file %d\n", k, count);
                    k++;
                    success++;
                }
                else
                {
                    printf("Block %d is already allocated\n", k);
                    k++;
                }
            }
            printf("\nFile Allocation Successful\n");
            printf("\nPress 1 to continue allocation, Press 0 to stop allocation: ");
            scanf("%d", &ch);
        }
        while (ch==1);
        printf("\nFile\tStart\tBlock Length\n");
        for (j=1 ; j<=count ; j++)
            printf("%2d%9d%10d\n", j, sb[j], b[j]);
}
```

OUTPUT:-

```
madhurya@madhurya:~$ gcc linked.c
madhurya@madhurya:~$ ./a.out

Enter the number of blocks already allocated: 5

Enter the already allocated blocks:
1
3
5
7
9

Enter the number of blocks occupied by file 1: 5

Enter the starting block of file 1: 1
Block 1 is already allocated
Block 2 has been allocated for file 1
Block 3 is already allocated
Block 4 has been allocated for file 1
Block 5 is already allocated
Block 6 has been allocated for file 1
Block 7 is already allocated
Block 8 has been allocated for file 1
Block 9 is already allocated
Block 10 has been allocated for file 1

File Allocation Successful

Press 1 to continue allocation, Press 0 to stop allocation: 0

File    Start   Block Length
 1        1         5
```

SOURCE CODE:-

```
//Indexed File Allocation
#include<stdio.h>
#include<stdlib.h>
void main()
{
	int i, j, n, mem[100], index, b[20], block[20], count=0, ib[20], ch;
	for (i=0 ; i<100 ; i++)
	{
		mem[i]=0;
		ib[i]=0;
	}
	for (i=0 ; i<20 ; i++)
	{
		block[i]=0;
		b[i]=0;
	}
	do
	{
		count++;
		printf("\nEnter the index block of the file %d: ", count);
		x:
		scanf("%d", &index);
		ib[count]=index;
		if (mem[index] == 0)
		{
			mem[index]=1;
			printf("Block %d has been allocated as index block for file %d\n", index,
			count);
		}
		else
		{
```

```
                printf("\nBlock %d is already allocated\n", index);
                printf("\nEnter another block for index: ");
                goto x;
            }
            printf("\nEnter the number of blocks occupied by file %d: ", count);
            scanf("%d", &b[count]);
            printf("\nEnter the blocks:\n");
            for (i=0 ; i<b[count] ; i++)
            {
                y:
                scanf("%d", &block[i]);
                if (mem[block[i]]==0)
                    mem[block[i]]=block[i];
                else
                {
                    printf("\nBlock %d is already allocated\n", block[i]);
                    printf("\nEnter another block: ");
                    goto y;
                }
            }
            printf("\nFile Allocation Successful\n");
            printf("\nPress 1 to continue allocation, Press 0 to stop allocation: ");
            scanf("%d", &ch);
        }
        while (ch==1);
        printf("\nFile\tIndex\tBlock Length\n");
        for (j=1 ; j<=count ; j++)
            printf("%2d%9d%10d\n", j, ib[j], b[j]);
}
```

OUTPUT:-

```
madhurya@madhurya:~$ gcc indexed.c
madhurya@madhurya:~$ ./a.out

Enter the index block of the file 1: 3
Block 3 has been allocated as index block for file 1

Enter the number of blocks occupied by file 1: 4

Enter the blocks:
1
2
5
7

File Allocation Successful

Press 1 to continue allocation, Press 0 to stop allocation: 1

Enter the index block of the file 2: 4
Block 4 has been allocated as index block for file 2

Enter the number of blocks occupied by file 2: 2

Enter the blocks:
5

Block 5 is already allocated
```

```
Enter another block: 6
8

File Allocation Successful

Press 1 to continue allocation, Press 0 to stop allocation: 0

File    Index   Block Length
 1        3         4
 2        4         2
madhurya@madhurya:~$
```

SOURCE CODE:-

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void passOne(char label[10], char opcode[10], char operand[10], char code[10], char mnemonic[3]);
void display();
int main()
{
	char label[10], opcode[10], operand[10];
	char code[10], mnemonic[3];
	passOne(label, opcode, operand, code, mnemonic);
	return 0;
}
void passOne(char label[10], char opcode[10], char operand[10], char code[10], char mnemonic[3])
{
	int locctr, start, length;
	FILE *fp1, *fp2, *fp3, *fp4, *fp5;
	fp1 = fopen("input3.txt", "r");
	fp2 = fopen("optab3.txt", "r");
	fp3 = fopen("symtab3.txt", "w");
	fp4 = fopen("intermediate3.txt", "w");
	fp5 = fopen("length3.txt", "w");
	fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
	if (strcmp(opcode, "START") == 0)
	{
		start = atoi(operand);
		locctr = start;
		fprintf(fp4, "\t%s\t%s\t%s\n", label, opcode, operand);
		fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
	}
	 else
		locctr = 0;
```

```
 while (strcmp(opcode, "END") != 0)
 {
        fprintf(fp4, "%d\t%s\t%s\t%s\n", locctr, label, opcode, operand);
        if (strcmp(label, "**") != 0)
        {
                fprintf(fp3, "%s\t%d\n", label, locctr);
        }
        fscanf(fp2, "%s\t%s", code, mnemonic);
        while (strcmp(code, "END") != 0)
        {
                if (strcmp(opcode, code) == 0)
                {
                        locctr += 3;
                        break;
                }
                fscanf(fp2, "%s\t%s", code, mnemonic);
        }
        if (strcmp(opcode, "WORD") == 0)
                locctr += 3;
        else if (strcmp(opcode, "RESW") == 0)
                locctr += (3 * (atoi(operand)));
        else if (strcmp(opcode, "BYTE") == 0)
                ++locctr;
        else if (strcmp(opcode, "RESB") == 0)
                locctr += atoi(operand);
    fscanf(fp1, "%s\t%s\t%s", label, opcode, operand);
 }
 fprintf(fp4, "%d\t%s\t%s\t%s\n", locctr, label, opcode, operand);
 fclose(fp4);
 fclose(fp3);
 fclose(fp2);
 fclose(fp1);
```

```
        length = locctr - start;
        fprintf(fp5, "%d", length);
        fclose(fp5);
        printf("\nThe length of the code : %d\n", length);
}
```

OUTPUT:-

```
madhurya@madhurya:~$ gcc pass1.c
madhurya@madhurya:~$ ./a.out

The length of the code : 12
```

input3.txt

```
**      START   1000
**      LDA     ALPHA
**      STA     BETA
ALPHA   RESW    1
BETA    RESW    1
**      END     **
```

intermediate3.txt

```
        **      START   1000
1000    **      LDA     ALPHA
1003    **      STA     BETA
1006    ALPHA   RESW    1
1009    BETA    RESW    1
1012    **      END     **
```

optab3.txt

```
LDA     00
STA     23
LDCH    15
STCH    18
END     *
```

symtab3.txt

```
ALPHA   1006
BETA    1009
```

length3.txt

```
12
```

SOURCE CODE:-

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
int main()
{
	FILE *fint,*ftab,*flen,*fsym;
	int i,len;
	char add[5],symadd[5],op[5],start[10], label[20],mne[10],operand[10],symtab[10],opmne[10];
	fint=fopen("input2.txt","r");
	flen=fopen("length2.txt","r");
	ftab=fopen("optab2.txt","r");
	fsym=fopen("symbol2.txt","r");
	fscanf(fint,"%s%s%s%s",add,label,mne,operand);
	if(strcmp(mne,"START")==0)
	{
		strcpy(start,operand);
		fscanf(flen,"%d",&len);
	}
	printf("H^%s^%s^%d\nT^00%s^",label,start,len,start);
	fscanf(fint,"%s%s%s%s",add,label,mne,operand);
	while(strcmp(mne,"END")!=0)
	{
		fscanf(ftab,"%s%s",opmne,op);
		while(!feof(ftab))
		{
			if(strcmp(mne,opmne)==0)
			{
				fclose(ftab);
				fscanf(fsym,"%s%s",symadd,symtab);
				while(!feof(fsym))
```

```
                {
                        if(strcmp(operand,symtab)==0)
                        {
                                printf("%s%s^",op,symadd);
                                break;
                        }
                        else
                                fscanf(fsym,"%s%s",symadd,symtab);
                }
                break;
        }
        else
                fscanf(ftab,"%s%s",opmne,op);
    }
    if((strcmp(mne,"BYTE")==0)||(strcmp(mne,"WORD")==0))
    {
        if(strcmp(mne,"WORD")==0)
                printf("0000%s^",operand);
        else
        {
                len=strlen(operand);
                for(i=2;i<len;i++)
                {
                        printf("%d",operand[i]);
                }
                printf("^");
        }
    }
    fscanf(fint,"%s%s%s%s",add,label,mne,operand);
    ftab=fopen("optab.txt","r");
    fseek(ftab,SEEK_SET,0);
}
```

```
        printf("\nE^00%s",start);
        fclose(fint);
        fclose(ftab);
        fclose(fsym);
        fclose(flen);
}
```

OUTPUT:-

input2.txt

```
-       COPY    START   1000
1000    -       LDA     ALPHA
1003    -       ADD     ONE
1006    -       SUB     TWO
1009    -       STA     BETA
1012    ALPHA   BYTE    C'KLNCE
1017    ONE     RESB    2
1019    TWO     WORD    5
1022    BETA    RESW    1
1025    -       END     -
```

optab2.txt

```
LDA     00
STA     23
ADD     01
SUB     05
```

symbol2.txt

```
1012    ALPHA
1017    ONE
1019    TWO
1022    BETA
```

length2.txt

```
25
```

```
madhurya@madhurya:~$ gcc pass2.c
madhurya@madhurya:~$ ./a.out
H^COPY^1000^25
T^001000^001012^231022^7576786769^00005^
E^001000
madhurya@madhurya:~$
```

SOURCE CODE:-

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
	FILE *fp;
	int i,addr1,l,j,staddr1;
	char name[10],line[50],name1[10],addr[10],rec[10],ch,staddr[10];
	printf("enter program name:" );
	scanf("%s",name);
	fp=fopen("objectcode.txt","r");
	fscanf(fp,"%s",line);
	for(i=2,j=0;i<8,j<6;i++,j++)
	{
		name1[j]=line[i];
	}
	name1[j]='\0';
	printf("name from obj. %s\n",name1);
	if(strcmp(name,name1)==0)
	{
		fscanf(fp,"%s",line);
		do
		{
			if(line[0]=='T')
			{
				for(i=2,j=0;i<8,j<6;i++,j++)
					staddr[j]=line[i];
				staddr[j]='\0';
				staddr1=atoi(staddr);
				i=12;
				while(line[i]!='$')
```

```
                        {
                                if(line[i]!='^')
                                {
                                        printf("00%d \t %c%c\n", staddr1,line[i],line[i+1]);
                                        staddr1++;
                                        i=i+2;
                                }
                                else
                                {
                                        i++;
                                }
                        }
                }
                else if(line[0]=='E')
                {
                        fclose(fp);
                }
                fscanf(fp,"%s",line);
        }
        while(!feof(fp) );
    }
}
```

OUTPUT:-

```
madhurya@madhurya:~$ gcc ablo.c
madhurya@madhurya:~$ ./a.out
enter program name:SAMPLE
name from obj. SAMPLE
001000   00
001001   10
001002   03
001003   07
001004   10
001005   09
002000   11
002001   11
002002   11
madhurya@madhurya:~$
```

objectcode.txt

```
H^SAMPLE^001000^0035
T^001000^0C^001003^071009$
T^002000^03^111111$
E^001000
```

SOURCE CODE:-

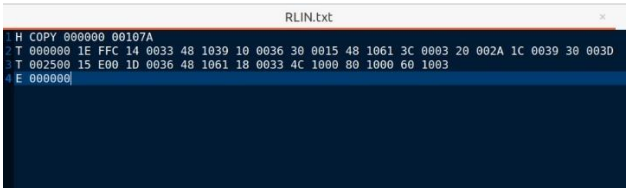
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void convert(char h[12]);
char bitmask[12];
char bit[12]={0};
void main()
{
	char add[6],length[10],input[10],binary[12],relocbit,ch,pn[5];
	int start,inp,len,i,address,opcode,addr,actualadd,tlen;
	FILE *fp1,*fp2;
	printf("\nEnter the actual starting address : ");
	scanf("%x",&start);
	fp1=fopen("RLIN.txt","r");
	fp2=fopen("RLOUT.txt","w");
	fscanf(fp1,"%s",input);
	fprintf(fp2," ----------\n");
	fprintf(fp2," ADDRESS\tCONTENT\n");
	fprintf(fp2," ----------\n");
	while(strcmp(input,"E")!=0)
	{
		if(strcmp(input,"H")==0)
		{
			fscanf(fp1,"%s",pn);
			fscanf(fp1,"%s",add);
			fscanf(fp1,"%s",length);
			fscanf(fp1,"%s",input);
		}
		if(strcmp(input,"T")==0)
		{
			fscanf(fp1,"%x",&address);
```

```
                        fscanf(fp1,"%x",&tlen);
                        fscanf(fp1,"%s",bitmask);
                        address+=start;
                        convert(bitmask);
                        len=strlen(bit);
                        if(len>=11)
                        len=10;
                        for(i=0;i<len;i++)
                        {
                                fscanf(fp1,"%x",&opcode);
                                fscanf(fp1,"%x",&addr);
                                relocbit=bit[i];
                                if(relocbit=='0')
                                        actualadd=addr;
                                else
                                        actualadd=addr+start;
                                fprintf(fp2,"\n %x\t\t%x%x\n",address,opcode,actualadd);
                                address+=3;
                        }
                        fscanf(fp1,"%s",input);
                }
        }
        fprintf(fp2," ----------------------------\n");
        fclose(fp2);
        fclose(fp1);
        printf("Successfully implemented relocating loader.\n");
}
void convert(char h[12])
{
        int i,l;
        strcpy(bit,"");
        l=strlen(h);
```

```
for(i=0;i<l;i++)
{
        switch(h[i])
        {
                case '0':
                        strcat(bit,"0");
                        break;
                case '1':
                        strcat(bit,"1");
                        break;
                case '2':
                        strcat(bit,"10");
                        break;
                case '3':
                        strcat(bit,"11");
                        break;
                case '4':
                        strcat(bit,"100");
                        break;
                case '5':
                        strcat(bit,"101");
                        break;
                case '6':
                        strcat(bit,"110");
                        break;
                case '7':
                        strcat(bit,"111");
                        break;
                case '8':
                        strcat(bit,"1000");
                        break;
                case '9':
```

```
                        strcat(bit,"1001");
                        break;
                    case 'A':
                        strcat(bit,"1010");
                        break;
                    case 'B':
                        strcat(bit,"1011");
                        break;
                    case 'C':
                        strcat(bit,"1100");
                        break;
                    case 'D':
                        strcat(bit,"1101");
                        break;
                    case 'E':
                        strcat(bit,"1110");
                        break;
                    case 'F':
                        strcat(bit,"1111");
                        break;
                }
            }
}
```

OUTPUT:-



RLOUT.txt

```
----------------------------
ADDRESS      CONTENT
----------------------------

4000         144033

4003         485039

4006         104036

4009         304015

400c         485061

400f         3c4003

4012         20402a

4015         1c4039

4018         30403d

401b         30403d

6500         1d4036

6503         485061

6506         184033

6509         4c1000

650c         801000

650f         601003
----------------------------
```

SOURCE CODE:-

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
void main()
{
	FILE *f1,*f2,*f3,*f4,*f5;
	int len,i,pos=1;
	char arg[20],mne[20],opnd[20],la[20],name[20],mne1[20],opnd1[20],pos1[10],pos2[10];
	f1=fopen("inputm.txt","r");
	f2=fopen("namtab.txt","w+");
	f3=fopen("deftab.txt","w+");
	f4=fopen("argtab.txt","w+");
	f5=fopen("op.txt","w+");
	fscanf(f1,"%s%s%s",la,mne,opnd);
	while(strcmp(mne,"END")!=0)
	{
		if(strcmp(mne,"MACRO")==0)
		{
			fprintf(f2,"%s\n",la);
			fseek(f2,SEEK_SET,0);
			fprintf(f3,"%s\t%s\n",la,opnd);
			fscanf(f1,"%s%s%s",la,mne,opnd);
			while(strcmp(mne,"MEND")!=0)
			{
				if(opnd[0]=='&')
				{
					sprintf(pos1,"%d",pos);
					strcpy(pos2,"?");
					strcpy(opnd,strcat(pos2,pos1));
					pos=pos+1;
				}
				fprintf(f3,"%s\t%s\n",mne,opnd);
```

```
                fscanf(f1,"%s%s%s",la,mne,opnd);
            }
            fprintf(f3,"%s",mne);
        }
        else
        {
            fscanf(f2,"%s",name);
            if(strcmp(mne,name)==0)
            {
                len=strlen(opnd);
                for(i=0;i<len;i++)
                {
                    if(opnd[i]!=',')
                        fprintf(f4,"%c",opnd[i]);
                    else
                        fprintf(f4,"\n");
                }
                fseek(f3,SEEK_SET,0);
                fseek(f4,SEEK_SET,0);
                fscanf(f3,"%s%s",mne1,opnd1);
                fprintf(f5,".\t%s\t%s\n",mne1,opnd);
                fscanf(f3,"%s%s",mne1,opnd1);
                while(strcmp(mne1,"MEND")!=0)
                {
                    if((opnd[0]=='?'))
                    {
                        fscanf(f4,"%s",arg);
                        fprintf(f5,"-\t%s\t%s\n",mne1,arg);
                    }
                    else
                        fprintf(f5,"-\t%s\t%s\n",mne1,opnd1);
                    fscanf(f3,"%s%s",mne1,opnd1);
                }
            }
```

```
                    else
                        fprintf(f5,"%s\t%s\t%s\n",la,mne,opnd);
            }
            fscanf(f1,"%s%s%s",la,mne,opnd);
        }
        fprintf(f5,"%s\t%s\t%s",la,mne,opnd);
        fclose(f1);
        fclose(f2);
        fclose(f3);
        fclose(f4);
        fclose(f5);
        printf("Successfully implemented single pass macroprocessor. \n");
}
```

OUTPUT:-

```
madhurya@madhurya:~$ gcc macro.c
madhurya@madhurya:~$ ./a.out
Successfully implemented single pass macroprocessor.
```

inputm.txt

```
SAMPLE START   1000
.      EX1     N1,N2
-      LDA     ?1
-      STA     ?2
N1     RESW    1
N2     RESW    1
-      END     -
```

op.txt

```
SAMPLE START   1000
.      EX1     N1,N2
-      LDA     ?1
-      STA     ?2
N1     RESW    1
N2     RESW    1
-      END     -
```

namtab.txt

```
EX1
```

namtab.txt

```
EX1
```

deftab.txt

```
EX1     &A,&B
LDA     ?1
STA     ?2
MEND
```

EXP. NO:- 10 **DATE:- 25/11/2022**

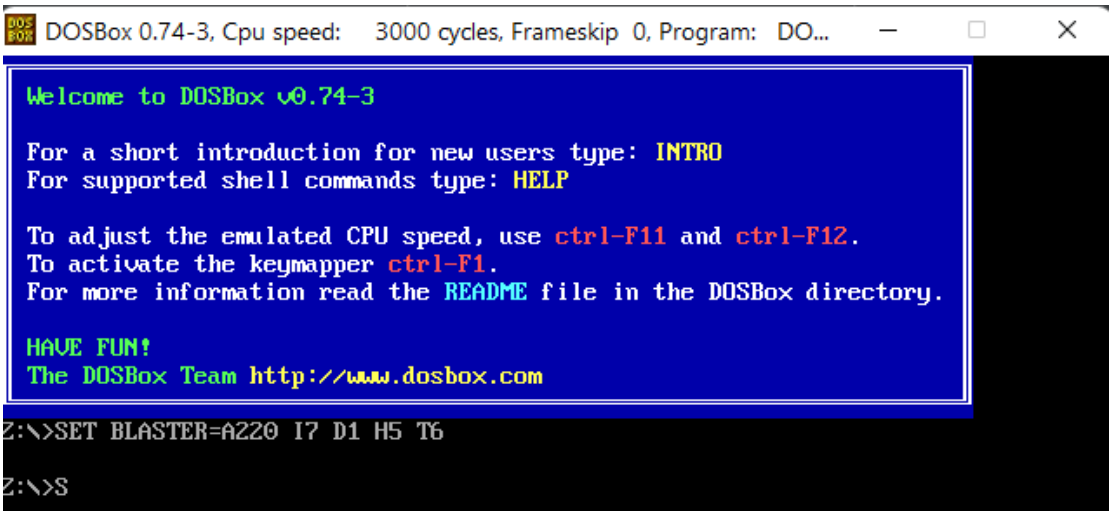
DEBUGGING COMMANDS

AIM:-

Study of assembler and debugging commands

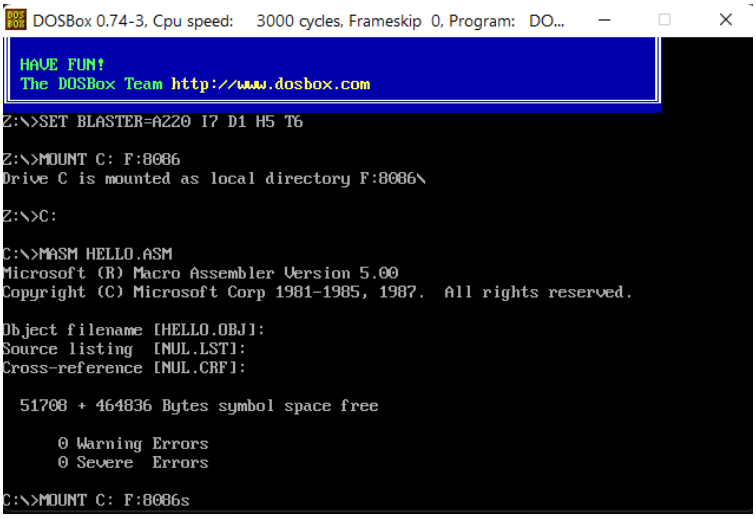
MASM ASSEMBLER

MASM is an assembler developed and maintained by Microsoft. MASM (8086) won't run on Windows 7. MASM cannot be run on newer versions of Windows, so we use it in DOSBOX Emulator.



We can use MASM compiler (masm.exe) inside DOSBOX by mounting the directory of 8086 MASM assembler by using the commands given below, inside DOSBOX

1) *MOUNT C: "MASM ASSEMBLER PATH"*
2) *C:*
3) *MASM "FILENAME.ASM"*



Save the source code (asm file) in the same directory as MASM assembler.

Example Code:

<u>Code to print “hello”:-</u>

```
DATA SEGMENT
        MSG DB 'HELLO_!$'
DATA ENDS
CODE SEGMENT
        ASSUME CS:CODE,DS:DATA
        START:
                MOV AX,DATA
                MOV DS,AX
                LEA DX,MSG
                MOV AH,9H
                INT 21h
                MOV AH,4CH
                INT 21H
CODE ENDS
END START
```

The assembler produces an object code (Here, hello.obj), which we can link using

```
LINK HELLO.OBJ
```

This creates the executable file named Hello.exe. It can be run and output will be produced.

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Program:  DO...
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [HELLO.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51758 + 464786 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>LINK HELLO.OBJ

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [HELLO.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>HELLO.EXE
HELLO_!
C:\>s
```

DEBUGGING

Debugging the executable file can be achieved by

DEBUG HELLO.EXE

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Program:  DE...
C:\>DEBUG HELLO.EXE
-T

AX=076A  BX=0000  CX=0021  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=075A  ES=075A  SS=0769  CS=076B  IP=0003   NV UP EI PL NZ NA PO NC
076B:0003 8ED8          MOV     DS,AX
-T

AX=076A  BX=0000  CX=0021  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=0005   NV UP EI PL NZ NA PO NC
076B:0005 8D160000      LEA     DX,[0000]                        DS:0000=4548
-T

AX=076A  BX=0000  CX=0021  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=0009   NV UP EI PL NZ NA PO NC
076B:0009 B409          MOV     AH,09
-T

AX=096A  BX=0000  CX=0021  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=000B   NV UP EI PL NZ NA PO NC
076B:000B CD21          INT     21
-G
HELLO_!
Program terminated normally
-s_
```

In debug mode, we can see the register values as well as the outputs that are printed to screen. This is useful for debugging any logical or syntax error in source code. Inside debug mode we can use commands to perform specified actions like t for trace etc.

assemble	A [address]
compare	C range address
dump	D[B|W|D] [range]
dump interrupt DI interrupt [count]	
dump LDT	DL selector [count]
dump MCB chain	DM
dump ext memory DX [physical_address]	
enter	E address [list]
fill	F range list
go	G [=address] [breakpoints]
hex add/sub	H value1 value2
input	I[W|D] port
load file	L [address]
load sectors	L address drive sector count
move	M range address
set x86 mode	M [x] (x=0..6)
set FPU mode	MC [2|N] (2=287,N=no FPU)
name	N [[drive:][path]filename [arglist]]
output	O[W|D] port value
proceed	P [=address] [count]
proceed return	PR
quit	Q
register	R [register [value]]
MMX register	RM
FPU register	RN[R]
toggle 386 regs RX	
search	S range list
trace	T [=address] [count]
trace mode	TM [0|1]
unassemble	U [range]
write file	W [address]
write sectors	W address drive sector count

prompts: '-' = real/v86-mode; '#' = protected-mode

RESULT:-

The study of assembler and debugging commands was completed successfully.

EXP. NO:- 11 **DATE:- 25/11/2022**

MASM – DECIMAL ARITHMETIC OPERATIONS

AIM:-

Write a MASM program to implement decimal arithmetic operations (16 bit)

a) Addition

b) Subtraction

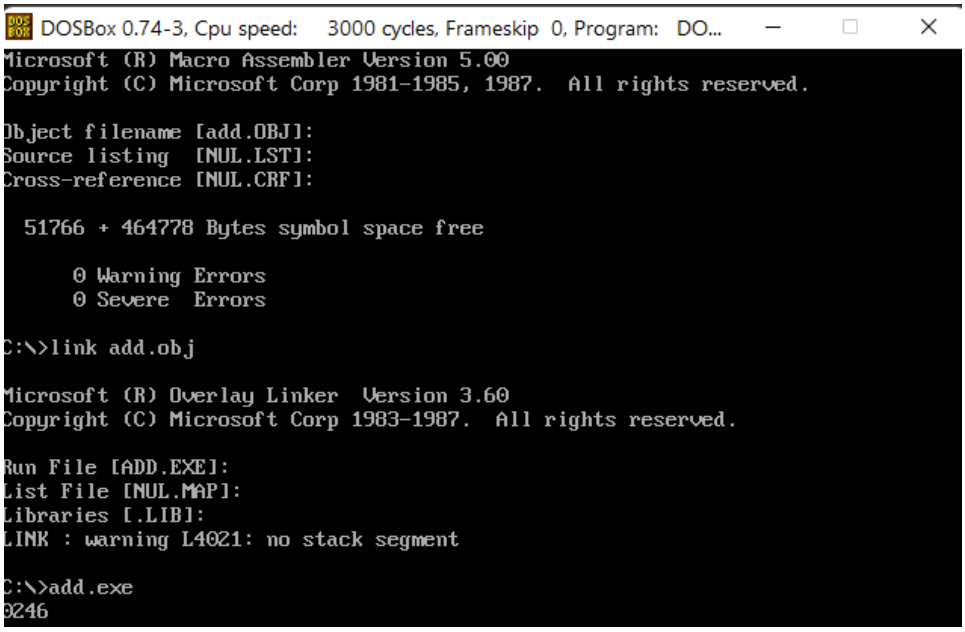
a) Addition

SOURCE CODE:-

```
DATA SEGMENT
    N1 DW 123H
    N2 DW 123H
DATA ENDS

STACK SEGMENT

STACK ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

    PRINT PROC
        MOV CH, 04H
        MOV CL, 04H
        L2:
            ROL BX, CL
            MOV DL, BL
            AND DL, 0FH
            CMP DL, 09
            JBE L4
            ADD DL, 07
        L4:
            ADD DL, 30H
            MOV AH, 02H
            INT 21H
            DEC CH
            JNZ L2
    RET
    PRINT ENDP

    START:
```

```
        MOV AX, DATA
        MOV DS, AX
        MOV AX, N1
        MOV BX, N2
        ADD AX, BX
        MOV BX, AX
        CALL PRINT
        JNC STOP
        MOV BX, 1
        CALL PRINT
    STOP:
        MOV AH, 4CH
        INT 21H
CODE ENDS
END START
```

OUTPUT:-



b) Subtraction

SOURCE CODE:-

```
DATA SEGMENT
    N1 DW 1234H
    N2 DW 234H
DATA ENDS

STACK SEGMENT

STACK ENDS

CODE SEGMENT
    ASSUME CS:CODE, DS:DATA

    PRINT PROC
        MOV CH, 04H
        MOV CL, 04H
        L2:
            ROL BX, CL
            MOV DL, BL
            AND DL, 0FH
            CMP DL, 09
            JBE L4
            ADD DL, 07
        L4:
            ADD DL, 30H
            MOV AH, 02H
            INT 21H
            DEC CH
            JNZ L2
    RET
    PRINT ENDP

    START:
        MOV AX, DATA
        MOV DS, AX
        MOV AX, N1
        MOV BX, N2
        SUB AX, BX
        MOV BX, AX
        CALL PRINT
        JNC STOP
        MOV BX, 1
        CALL PRINT
    STOP:
        MOV AH, 4CH
        INT 21H
CODE ENDS
END START
```

OUTPUT:-

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Program:  DO...
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [sub.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51766 + 464778 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link sub.obj

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [SUB.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>sub.exe
1000
```

RESULT:-

The program was successfully executed and output was verified.

EXP. NO:- 12 **DATE:- 09/12/2022**

MASM – STRING MANIPULATION

AIM:-

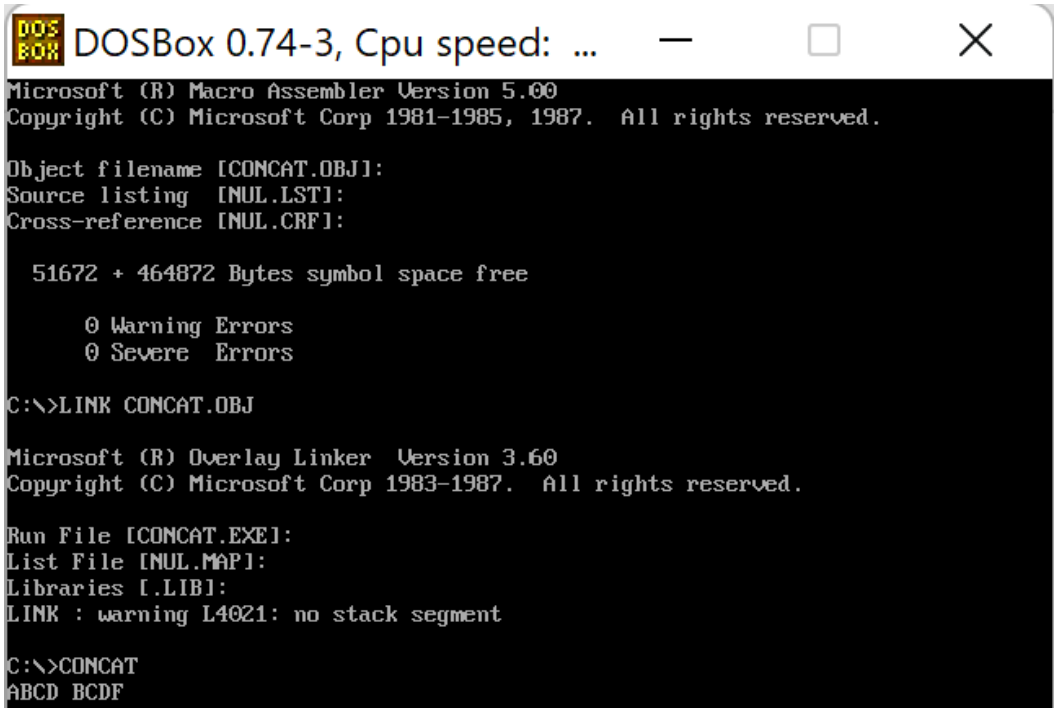
Write a MASM program to implement string manipulation

a) Concatenation

SOURCE CODE:-

```
DATA SEGMENT
    STRING DB "ABCD $"
    STRING2 DB "BCDF $"
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
    START:
        MOV AX,DATA
        MOV DS,AX
        LEA SI,STRING
        LEA DI,STRING2
        MOV AL,'$'
    LP:   CMP AL,[SI]
        JZ NEXT
        INC SI
        JMP LP
    NEXT:  CMP AL,[DI]
        JZ EXIT
        MOV AH,[DI]
        MOV [SI],AH
        INC DI
        INC SI
        JMP NEXT
    EXIT:  LEA DX,STRING
        MOV AH,09H
        INT 21H
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

OUTPUT:-



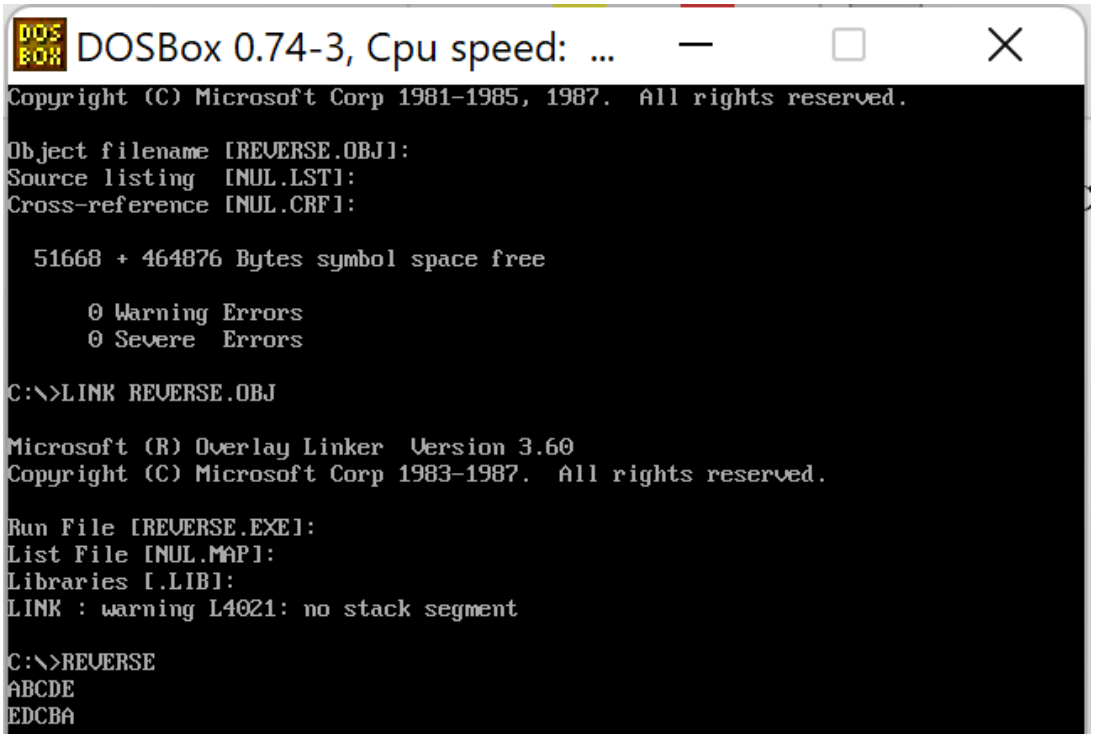
b) Reversal

SOURCE CODE:-

```
DATA SEGMENT
    STR1 DB 100 DUP('?')
    STR2 DB 100 DUP('?')
DATA ENDS

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
    START:
        MOV AX,DATA
        MOV DS,AX
        MOV BL,01H
        LEA SI,STR1
        LEA DI,STR2
        MOV AH,01H
        INT 21H
        MOV [SI],AL
        INC SI
    INPUT:  CMP AL,13
        JZ EXIT
        MOV [SI],AL
        INC SI
        INC BL
        MOV AH,01H
        INT 21H
```

```
        JMP INPUT
EXIT:   DEC SI
        MOV CL,BL
        DEC CL
LP:     MOV DL,[SI]
        MOV [DI],DL
        DEC SI
        INC DI
        LOOP LP
        MOV AL,'$'
        MOV [DI],AL
        LEA DX,STR2
        MOV AH,09H
        INT 21H
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

OUTPUT:-



RESULT:-

The program was successfully executed and output was verified.

EXP. NO:- 13 **DATE:- 09/12/2022**

MASM - SEARCHING

AIM:-

Write a MASM program to implement searching of 16 bit numbers

SOURCE CODE:-

```
DATA SEGMENT
    MSG1 DB "ITEM FOUND","$"
    ARRAY DB 0015H,0020H,0010H,0003H,0045H
    MSG2 DB "NOT FOUND","$"
DATA ENDS

PRINT MACRO MSG
    MOV AH,09H
    LEA DX,MSG
    INT 21H
    INT 3
ENDM

CODE SEGMENT
    ASSUME CS:CODE,DS:DATA
    START:
        MOV AX,DATA
        MOV DS,AX
        MOV CL,05H
        MOV AX,0003H
        LEA SI,ARRAY
    LP:  MOV BL,[SI]
        CMP AX,BX
        JNE L2
        PRINT MSG1
        MOV AH,4CH
        INT 21H
    L2:  INC SI
        DEC CL
        CMP CL,0
        JNE LP
        PRINT MSG2
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

OUTPUT:-

Item being searched is 0003H and it is found in array

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Program:  DO...
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [SEARCH.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51670 + 464874 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>LINK SEARCH.OBJ

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [SEARCH.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>SEARCH.EXE
FOUND
```

Item being searched is 0033H and it is not found in array

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Program:  DO...
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [SEARCH.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51670 + 464874 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>LINK SEARCH.OBJ

Microsoft (R) Overlay Linker  Version 3.60
Copyright (C) Microsoft Corp 1983-1987.  All rights reserved.

Run File [SEARCH.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>SEARCH.EXE
NOT FOUND
```

RESULT:-

The program was successfully executed and output was verified.

EXP. NO:- 14 **DATE:- 09/12/2022**

MASM - SORTING

AIM:-

Write a MASM program to implement sorting of 16 bit numbers

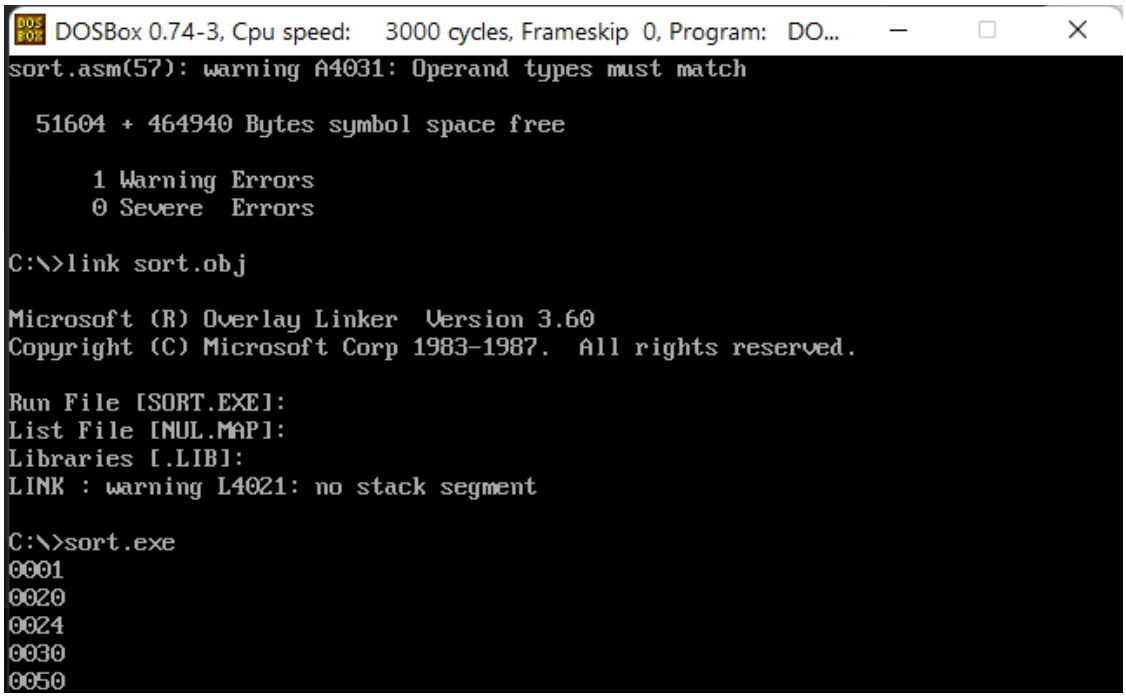
SOURCE CODE:-

```
DATA SEGMENT
        ARRAY DW 0050H,0024H,0001H,0020H,0030H,"$"
DATA ENDS
STACK SEGMENT
STACK ENDS
PRINT MACRO ST
        MOV AH,09H
        LEA DX, ST
        INT 21H
        INT 3
ENDM
INTEGER_PRINT MACRO X
        MOV CH,04H
        MOV CL,04H
        L1:
                ROL X,CL
                MOV DL,X
                AND DL,0FH
                CMP DL, 09
                JBE L2
                ADD DL,07
        L2:     ADD DL, 30H
                MOV AH,02H
```

```
            INT 21H
            DEC CH
            JNZ L1
            INT 3
ENDM
CODE SEGMENT
        ASSUME CS:CODE, DS:DATA
        START:
            MOV AX,DATA
            MOV DS,AX
            MOV BX,4
        SORT:   LEA SI, ARRAY
            MOV CX,BX
        ITR:    MOV AX,[SI]
            INC SI
            INC SI
            CMP AX,[SI]
            JC LESS
            XCHG AX,[SI]
            XCHG AX,[SI-2]
        LESS:   LOOP ITR
            DEC BX
            CMP BX,0
            JNZ SORT
            MOV DH,05
            LEA DI,ARRAY
        PRT:    MOV BX,[DI]
            INTEGER_PRINT BX
            MOV DL,0AH
            MOV AH,02H
            INT 21H
            INC DI
```

```
        INC DI
        DEC DH
        JNZ PRT
        MOV AH,4CH
        INT 21H
CODE ENDS
END START
```

OUTPUT:-



RESULT:-

The program was successfully executed and output was verified.

EXP. NO:- 15 **DATE:- 02/12/2022**

DECIMAL ARITHMETIC OPERATION (8086)

AIM:-

Write a program to implement simple decimal arithmetic operations in 8086 trainer kit Implement simple decimal arithmetic operations using 8086 trainer kit

a) Addition
b) Subtraction
c) Multiplication
d) Divison(32 bit / 16 bit)

a) Addition of two 8-bit numbers (with carry)

SOURCE CODE:-

MEMORY ADDRESS	OBJECT CODE	MNEMONICS
1000	8A 26 00 11	MOV AH,[1100]
1004	8A 1E 01 11	MOV BL,[1101]
1008	C6 C0 00	MOV AL,00
100B	00 DC	ADD AH,BL
100D	73 02	JNC 1011
100F	FE C0	INC AL
1011	88 06 02 11	MOV [1102],AL
1015	88 26 03 11	MOV [1103],AH
1019	F4	HLT

INPUT:-

1100=FF
1101=FF

OUTPUT:-

1102=01
1103=FE

b) Subtraction of two 16-bit numbers

SOURCE CODE:-

MEMORY ADDRESS	OBJECT CODE	MNEMONICS
1000	8A	MOV AX,[1101]
1001	06	
1002	00	
1003	11	
1004	2B	SUB AX,[1102]
1005	06	
1006	02	
1007	11	
1008	89	MOV [1200],AX
1009	06	
100A	00	
100B	12	
100C	F4	HLT

INPUT:-

Minuend: 1100=9999
Subtrahend : 1102=369C

OUTPUT:-

1200=62FD

c) Multiplication of two 16-bit numbers

SOURCE CODE:-

MEMORY ADDRESS	OBJECT CODE	MNEMONICS
1000	8B 06 00 11	MOV AX,[1100]
1004	8B 06 02 11	MOV BX,[1102]
1008	F7 E3	MUL BX
100A	89 06 00 12	MOV [1200],AX
100E	89 16 02 12	MOV [1202],DX
1012	F4	HLT

INPUT:-

1100=EF1A
1102=CD50

OUTPUT:-

1200=8A20
1202=BFC2

d) Division of 32-bit by 16-bit

SOURCE CODE:-

MEMORY ADDRESS	OBJECT CODE	MNEMONICS
1000	8B 06 06 11	MOV AX,[1100]
1004	2B 06 02 11	MOV DX,[1102]
1008	89 06 02 12	MOV BX,[1104]
100C	8B 06 04 11	DIV BX
1010	1B 06 00 11	MOV [1200],AX
1014	89 06 00 12	MOV [1202],DX
1018	F4	HLT

INPUT:-

Dividend : 1100= 580A
1102=71C2

Divisor :1104=F6F2

OUTPUT:-

Quotient: 1200 = 75EE
Remainder: 1202= 290E

RESULT:-

The program was successfully executed and output was verified.

EXP. NO:- 16 **DATE:- 02/12/2022**

CODE CONVERSION USING 8086

AIM:-

Implement code conversions

a) Hexadecimal to BCD

b) BCD to Hexadecimal

a) Hexadecimal to BCD

SOURCE CODE:-

MEMORY ADDRESS	OBJECT CODES	MNEMONICS
1000	8A 26 00 11	MOV AH,[1100]
1004	80 E4 F0	AND AH,F0
1007	B1 04	MOV CL,04
1009	B0 00	MOV AL,00
100B	D2 EC	SHR AH,CL
100D	80 FC 00	CMP AH,00
1010	74 0D	JE 101F
1012	B7 00	MOV BH,00
1014	04 16	ADD AL,016
1016	27	DAA
1017	73 02	JNC 101B
1019	FE C7	INC BH
101B	FE CC	DEC AH
101D	75 F5	JNZ 1014
101F	8A 26 00 11	MOV AH,[1100]
1023	80 E4 0F	AND AH,0F
1026	80 FC 0A	CMP AH,0A
1029	72 03	JC 102E
102B	80 C4 06	ADD AH,06
102E	00 E0	ADD AL,AH
1030	27	DAA
1031	73 02	JNC 1035
1033	FE C7	INC BH
1035	88 3E 01 11	MOV [1101],BH
1039	A2 02 11	MOV[1102],AL
103C	F4	HLT

INPUT:-

[1100]=99

OUTPUT:-

[1101]=01

[1102]=53

b) BCD to Hexadecimal

SOURCE CODE:-

MEMORY ADDRESS	OBJECT CODES	MNEMONICS
1000	A0 01 11	MOV AL,[1101]
1003	B1 00	MOV CL,00
1005	3C 16	CMP AL,016
1007	72 07	JB 1010
1009	2C 16	SUB AL,016
100B	2F	DAS
100C	FE C1	INC CL
100E	EB F5	JMP 1005
1010	3C 0A	CMP AL,0A
1012	72 02	JB 1016
1014	2C 06	SUB AL,06
1016	88 C3	MOV BL,AL
1018	B0 10	MOV AL,10
101A	F6 E1	MUL CL
101C	00 D8	ADD AL,BL
101E	A2 02 11	MOV[1102],AL
1021	F4	HLT

INPUT:-

[1101]=63

OUTPUT:-

[1102]=3F

RESULT:-

The program was successfully executed and output was verified.

EXP. NO:- 17 **DATE:- 02/12/2022**

SIMPLE ARITHMETIC OPERATIONS USING 8051

AIM:-

Familiarisation of 8051 Kit by executing simple arithmetic operations

a) Addition
b) Subtraction

a) Addition

SOURCE CODE:-

```
8000    90,81,00    MOV DPTR,#8100
8003    E0          MOVX A,@DPTR
8004    F8          MOV R0,A
8005    A3          INC DPTR
8006    E0          MOVX A, @DPTR
8007    28          ADD A,R0
8008    A3          INC DPTR
8009    F0          MOVX @DPTR,A
800A    74,00       MOV A,#00
800C    34,00       ADDC A,#00
800E    A3          INC DPTR
800F    F0          MOVX @DPTR,A
8010    02,80,10    HERE:LJUMP HERE
```

INPUT:-

[8100]=01

[8101]=02

OUTPUT:-

[8102]=03

[8103]=00

b) Subtraction

8000	90,81,00	MOV DPTR,#8100
8003	E0	MOVX A,@DPTR
8004	F8	MOV R0,A
8005	A3	INC DPTR
8006	E0	MOVX A, @DPTR
8007	A3	INC DPTR
8008	C3	CLR C
8009	98	SUBB A,R0
800A	F0	MOVX @DPTR,A
800B	A3	INC DPTR
800C	74,00	MOV A,#00
800E	38	ADDC A,#00
800F	F0	MOVX @DPTR,A
8010	02,80,11	HERE:LJUMP HERE

INPUT:-

[8100]=03

[8101]=05

OUTPUT:-

[8102]=02

[8103]=00

RESULT:-

The program was successfully executed and output was verified.