**SSN College of Engineering Department of Computer Science and Engineering**

**III year - UCS1512 – Microprocessors Lab**

**Matrix operations**

**Exp No:** 05

**Name:** Rahul Ram M

**Register Number:** 185001121

**Date:** 03/10/2020

**a) Matrix addition:**

**Aim:**

Design 8086 program for matrix addition.

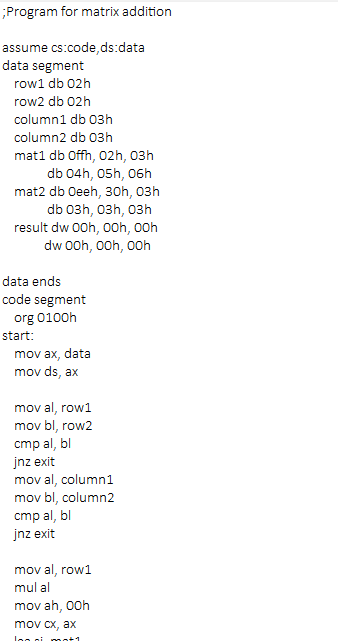
**Procedure for executing MASM:**

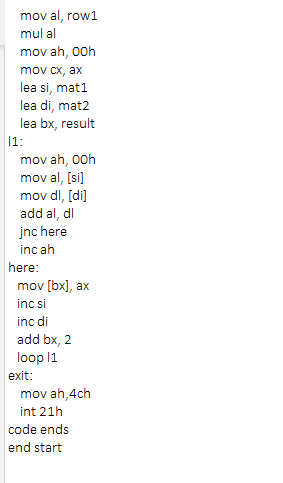
1. Run Dosbox and mount your masm folder to a drive in dosbox.
2. Goto the mounted drive.
3. Save the 8086 program with extension .asm in the same folder using command “edit”
4. After creating the file, assemble it using the command “masm filename.asm”
5. Link the file using the command “link filename.obj;”
6. Use debug command with filename.exe to execute and analyse the memory contents, “debug filename.exe”.
7. In debug, command “u” will display the unassembled code.
8. Use command “d segment:offset” to see the content of memory locations starting from segment:offset address.
9. To change the value in memory, use the command “e segment:offset”
10. Verify the memory contents to ensure the updates (using command “d”).
11. . Execute using the command “g” and check the outputs.
12. “q” to exit from debug and “exit” to exit from command prompt and to close the Dosbox.

**Algorithm:**

1. Move the starting address of data segment to AX register and move the data from AX register to DS register.
2. Move the value of the variable ROW1 to AL register and ROW2 to BL register.
3. Compare AL and BL register. IF they are not equal, jump to EXIT.
4. Move the value of the variable COLUMNN1 to AL register and COLUMN2 to BL register.
5. Compare AL and BL register. IF they are not equal, jump to EXIT.
6. Move the value of the variable ROW1 to AL register.
7. Multiply AL and BL using MUL AL.
8. Move 00H to AL register and copy AX to CX register.
9. Load effective addresses of the variables MAT1, MAT2 and RESULT to SI, Di and BX register respectively.
10. L1: Move 00H to AH register.
11. Move the value at SI and DI register’s location to AL and DL register respectively.
12. Now add AL and DL. If there is no carry, jump to HERE else increment AH register.
13. HERE: Move the value of AX register to BX’s location.
14. Increment SI and Di register and add 2 to BX register.
15. Loop to L1.
16. EXIT: INT 21H means invoke the interrupt identified by the hexadecimal number 21. In MS-DOS, invoking interrupt 21h while AH = 4Ch causes the current process to terminate and uses the value of register AL as the exit code of the process.

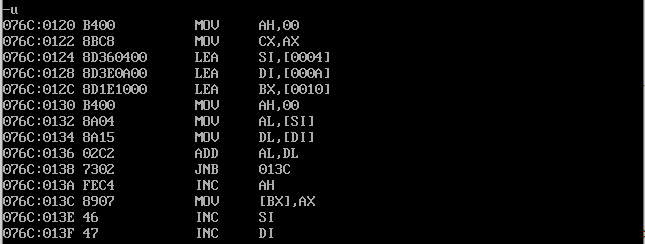
**Program:**





|  |  |  |
| --- | --- | --- |
|  | **Program** | **Comments** |
| START: | ORG 0100H | Memory instruction starts from 0010H. |
| MOV AX, DATA  MOV DS, AX | Transferring the data from DATA to AX register and  from AX register to DS register. |
| MOV AL, ROW1 | AL <- ROW1 |
| MOV BL, ROW2 | BL <- ROW2 |
| CMP AL, BL | Compare AL and BL register. |
| JNZ EXIT | Jump to EXIT if not equal. |
| MOV AL, COLUMN1 | AL <- COLUMN1 |
| MOV BL, COLUMN2 | BL <- COLUMN2 |
| CMP AL, BL | Compare AL and BL register. |
| JNZ EXIT | Jump to EXIT if not equal. |
| MOV AL, ROW1 | AL <- ROW1 |
| MUL AL | AX <- AL x BL |
| MOV AH, 00H | AH <- 00H |
| MOV CX, AX | CX <- AX |
| LEA SI, MAT1 | Load effective address of MAT1 to SI. |
| LEA DI, MAT2 | Load effective address of MAT2 to DI. |
| LEA BX. RESULT | Load effective address of RESULT to BX. |
| L1: | MOV AH, 00H | AH <- 00H |
| MOV AL, [SI] | AL <- [SI] |
| MOV DL, [DI] | DL <- [DI] |
| ADD AL, DL | AL <- AL + DL |
| JNC HERE | Jump to HERE if there is a carry. |
| INC AH | Increment AH. |
| HERE: | MOV [BX], AX | [BX] <- AX |
| INC SI | Increment SI. |
| INC DI | Increment DI. |
| ADD BX, 2 | BX <- BX + 2 |
| LOOP L1 | Loop to L1. |
| EXIT: | MOV AH, 4CH  INT 21H | Terminates the program. |

**Snapshot of sample input and output:**



**Adding matrices of same dimensions:**

MAT1 [[FF, 02, 03], [04, 05, 06]]

MAT2 [[EE, 30, 03], [03, 03, 03]]

RESULT [[01 ED, 00 32, 00 06], [00 07, 00 08, 00 09]]



**Adding matrices of different dimensions.**

MAT1 [[FF, 02, 03]]

MAT2 [[EE, 30, 03], [03, 03, 03]]

**Result:**

Thus the 8086 program for matrix addition is executed successfully in DOS-BOX.

**b) Matrix subtraction:**

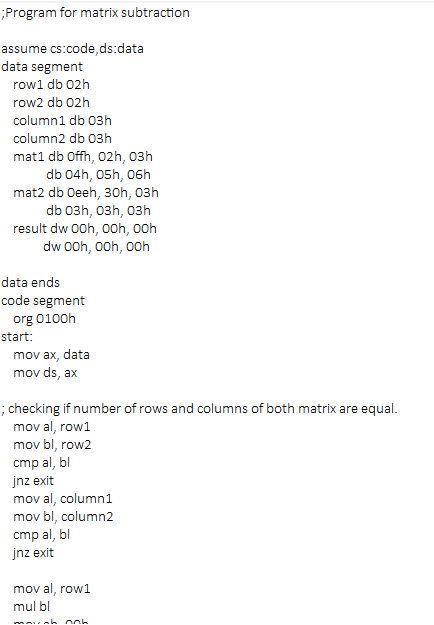
**Aim:**

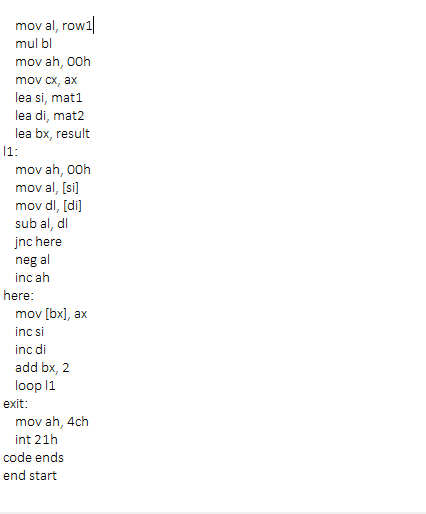
Design 8086 program for Matrix subtraction.

**Algorithm:**

1. Move the starting address of data segment to AX register and move the data from AX register to DS register.
2. Move the value of the variable ROW1 to AL register and ROW2 to BL register.
3. Compare AL and BL register. IF they are not equal, jump to EXIT.
4. Move the value of the variable COLUMNN1 to AL register and COLUMN2 to BL register.
5. Compare AL and BL register. IF they are not equal, jump to EXIT.
6. Move the value of the variable ROW1 to AL register.
7. Multiply AL and BL using MUL AL.
8. Move 00H to AL register and copy AX to CX register.
9. Load effective addresses of the variables MAT1, MAT2 and RESULT to SI, Di and BX register respectively.
10. L1: Move 00H to AH register.
11. Move the value at SI and DI register’s location to AL and DL register respectively.
12. Now sub AL and DL. If there is no carry, jump to HERE else negate AL and increment AH register.
13. HERE: Move the value of AX register to BX’s location.
14. Increment SI and Di register and add 2 to BX register.
15. Loop to L1.
16. EXIT: INT 21H means invoke the interrupt identified by the hexadecimal number 21. In MS-DOS, invoking interrupt 21h while AH = 4Ch causes the current process to terminate and uses the value of register AL as the exit code of the process.

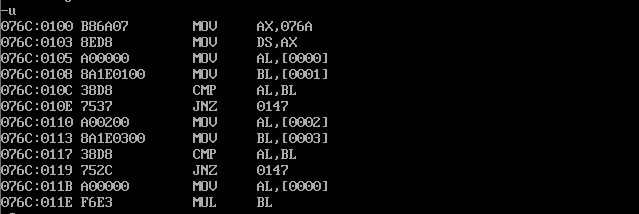
**Program:**





|  |  |  |
| --- | --- | --- |
|  | **Program** | **Comments** |
| START: | ORG 0100H | Memory instruction starts from 0010H. |
| MOV AX, DATA  MOV DS, AX | Transferring the data from DATA to AX register and  from AX register to DS register. |
| MOV AL, ROW1 | AL <- ROW1 |
| MOV BL, ROW2 | BL <- ROW2 |
| CMP AL, BL | Compare AL and BL register. |
| JNZ EXIT | Jump to EXIT if not equal. |
| MOV AL, COLUMN1 | AL <- COLUMN1 |
| MOV BL, COLUMN2 | BL <- COLUMN2 |
| CMP AL, BL | Compare AL and BL register. |
| JNZ EXIT | Jump to EXIT if not equal. |
| MOV AL, ROW1 | AL <- ROW1 |
| MUL AL | AX <- AL x BL |
| MOV AH, 00H | AH <- 00H |
| MOV CX, AX | CX <- AX |
| LEA SI, MAT1 | Load effective address of MAT1 to SI. |
| LEA DI, MAT2 | Load effective address of MAT2 to DI. |
| LEA BX. RESULT | Load effective address of RESULT to BX. |
| L1: | MOV AH, 00H | AH <- 00H |
| MOV AL, [SI] | AL <- [SI] |
| MOV DL, [DI] | DL <- [DI] |
| SUB AL, DL | AL <- AL - DL |
| JNC HERE | Jump to HERE if there is a carry. |
| NEG AL | Negate the value in the AL register (takes 2’s compliment). |
| INC AH | Increment AH. |
| HERE: | MOV [BX], AX | [BX] <- AX |
| INC SI | Increment SI. |
| INC DI | Increment DI. |
| ADD BX, 2 | BX <- BX + 2 |
| LOOP L1 | Loop to L1. |
| EXIT: | MOV AH, 4CH  INT 21H | Terminates the program. |

**Snapshot of sample input and output:**

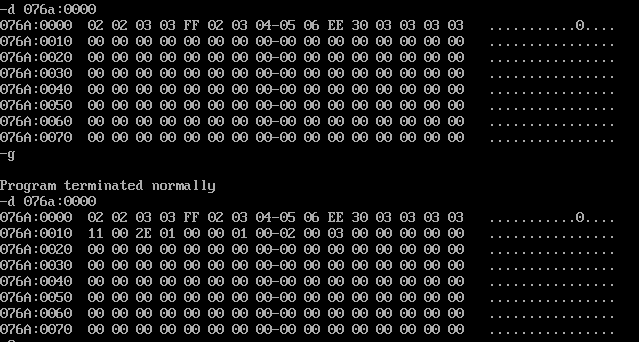


**Subtracting matrices of same dimensions:**

MAT1 [[FF, 02, 03], [04, 05, 06]]

MAT2 [[EE, 30, 03], [03, 03, 03]]

RESULT [[00 11, 01 2E, 00 00], [00 01, 00 02, 00 03]]



**Subtracting matrices of different dimensions:**

MAT1 [[FF, 02, 03]]

MAT2 [[EE, 30, 03], [03, 03, 03]



**Result:**

Thus the 8086 program for matrix subtraction is executed successfully in

DOS-BOX.