VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Fall Semester 2021-22

Microprocessors and Interfacing LAB

CSE2006

Slot – L43+L44

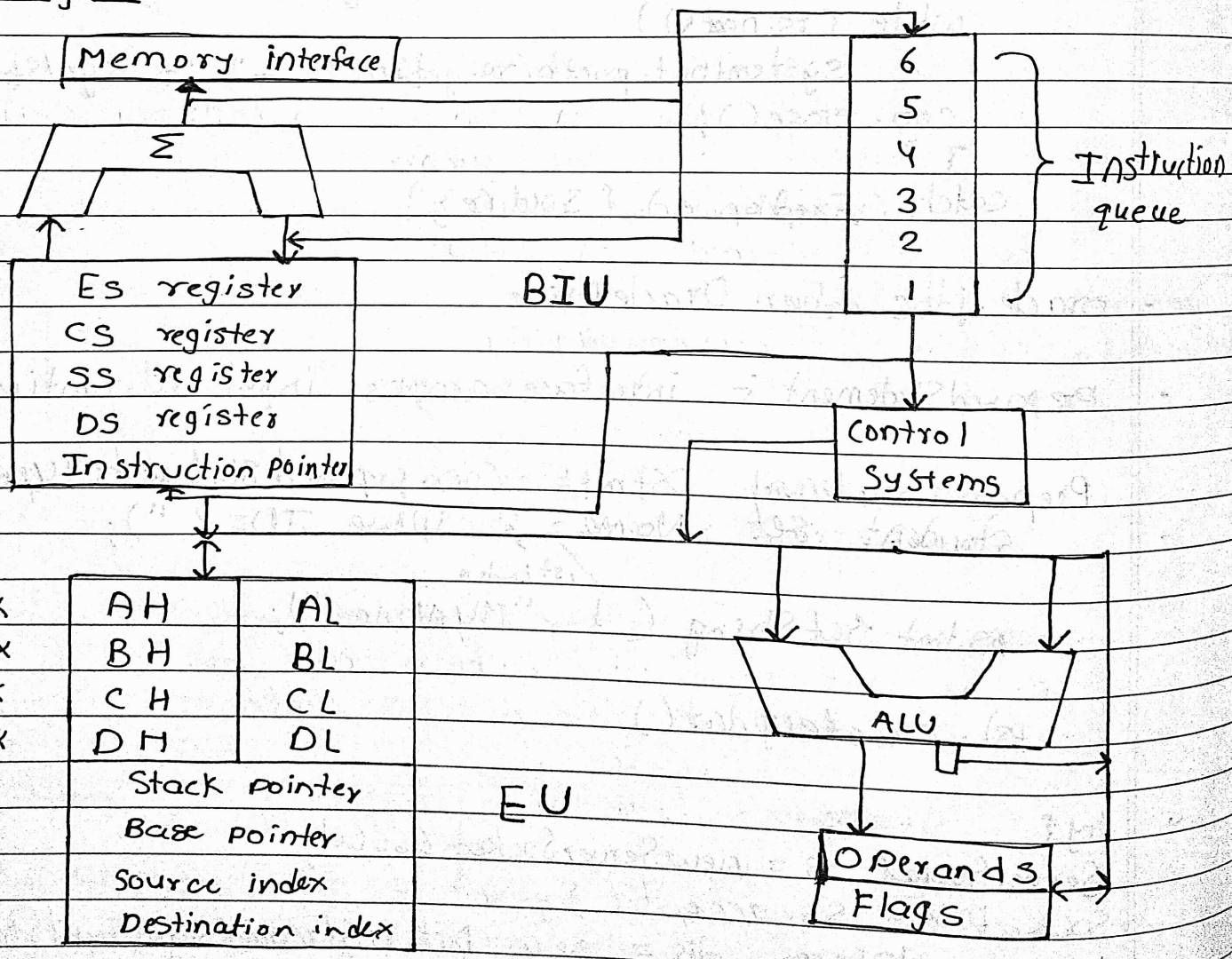
Digital Assignment 2

Name: Ishan Sagar Jogalekar

Reg. No: 19BCE2250

Q.1 Architecture of 8086 :-

Diagram :-



Memory Segmentation:

To increase execution speed & fetching speed 8086 segments the memory.

It divides into 2 units.

i) BUS interface unit (BIU) :- Bus control logic of BIU generates all the bus control signals such as read and write signals for memory and I/O.

- Handles all transfer of data & address on the buses for EU.

Function of BIU:-

- Fetch the instruction or data from memory.
- write data to memory.
- write data to port.
- Read data from port.

Contains 4 segment register.

i) Instruction pointer (IP):- Always point to next inst to be executed. Offset address is relative to CS. Always work together with CS.

ii) Segment register :-

a. CS - Points at segment containing current program
b. DS - General points at segment where variables are defined.
c. ES - Extra segment, it up to a code to define usage.
d. SS - points at segment containing Stack.

- Address generation circuit
- 6 byte pre-fetch queue

1. ii) Execution Unit :-

EU is also called a functional unit. It is part of CPU that performs the operations & calculations are instructed by computer program.

Functions :-

- To tell BIU where to fetch the inst or data from memory interface
- To decode the instructions.
- To execute the instructions.

General purpose registers :-

- AX - Accumulator register → Arithmatic logic & data transfer, Input & output
- BX - Base address register → Contain data pointer, used for data, based indexed.
- CX → Count register → Iterative code segments using loop inst. count of bits shift & rotate
- DX → Data register → used as port no. in I/O operation

ALU :-

- Stack pointer :- Always points to top item on Stack. offset address relative to SS.
- Base pointer :- Primarly used to access parameters passed via Stack.
- Source index :- used to pointer addressing of data
- Destination index :- can be used for pointer addressing of data.

19BCE2250- Ishan Jogalekar

1. Flag / status register :-

1. Carry flag 2. parity flag 3. Auxillary flag
4. Zero flag 5. Sign flag 6. overflow flag

3 control flag :-

1. Trap flag 2. Interrupt flag 3. Direction flag

Q.2 Pin description of 8086 :-

GND	1	40	VCC
AD14	2	39	AD15
AD13	3	38	A16/S3
AD12	4	37	A17/S4
AD11	5	36	A18/S5
AD10	6	35	A19/S6
AD9	7	34	$\overline{BHE}$/S7
AD8	8	33	MN/$\overline{MX}$
AD7	9	32	$\overline{RD}$
AD6	10	31	$\overline{RQ}$/$\overline{GT0}$ (Hold)
AD5	11	30	RQ/$\overline{GT_1}$ (HLDA)
AD4	12	29	$\overline{LOCK}$ ($\overline{WR}$)
AD3	13	28	$\overline{S2}$ (M/IO)
AD2	14	27	$\overline{S1}$ (DT/$\overline{R}$)
AD1	15	26	$\overline{S0}$ ($\overline{DEN}$)
AD0	16	25	QS0 (ALE)
NMI	17	24	QS1
INTR	18	23	$\overline{TEST}$
CLK	19	22	READY
GND	20	21	RESET

2. • AD0 - AD15 (I/O) :- Address data bus :-
low order bus, AD0 - AD7 carries low order data byte while AD8 - AD15 carry higher order byte. They are multiplexed with data, for memory address.

• AD16 - A19 / $S_0 - S_1 - S_2$:-
A16 - A19 are higher order address bus. These are multiplex with status signals. In case of memory operations these pins act as address bus.

S_2	S_1	S_0	characteristics
0	0	0	Interrupt ack
0	0	1	Read I/O port
0	1	0	write I/O port
0	1	1	Halt
1	0	0	code access
1	0	1	Read memory
1	1	0	write memory
1	1	1	Passive state

• S5 → Presense of interrupts in microprocessor, serves interiput flag.

• S6 → Status of bust master for current operation, more specifically bus master.

• BHE/S7 → Bus enable Status. used to enable data onto MSH of data bus, BHE uses low signal. It is multiplex with status signal S7.

- RD(O) :- READ → Indicates processor is performing a memory or I/O read cycle, active when low.
- TEST(I) → If Test pin is low, execution continues, Otherwise processor waits in 'idle' position or state.
- INTR (Interrupt request) → level triggered input which is sampled during last clock cycle of each inst to determine the interrput ack opration.
- NMI (non-maskable interrupt) :- Edge triggred input, cause type-2 interrupt, vector look up table located in system memory.
- Reset (I) :- Cause processor to immediately terminate it present activity. Signal is high for at least last 4 clock cycles.
- Ready (I) :- Acknowldgement from addressed memory or I/O device
- CLK (I) :- clock, Provide basic time for processer & bus controller.
- MN / MX (I) :- Indicates mode on which processor is working.
- WR(O) :- Indicates processor is performing a write memory or write I/O cycle.
- INTA(O) :- Interrupt acknowledge → use as read strobe for interrupt ack cycles.

2. • ALE(O) :- Address latch enable → provided by processor to latch address.

• DT/R(O) :- Data trasmit, use do to control direct of data flow.

• DEN (O) :- Data enable, provide as Output enable for 8286/8287 in a min system.

• HOLD & HLDA (I/O) :- HOLD indicated that another Master has been requesting a bus, Active high (1), it issues HLDA acknowledgment.

• QS0, QS1 (O) :- These signals indicate the status of internal 8086.

QS0	QS1	Status
0	0	No operation
1	0	First byte of OP code from queue
0	1	Empty queue
1	1	Subsequent byte from queue

• RQ/GT0 & RQ/GT1 (I/O) :- Request/Grant → Pins are used by other processor in multi processors orginization.

19BCE2250 - Ishan Jogalekar

Q.3 Addressing modes :-

The way in which the processor gets data from users is called addressing modes.
It can get data from different sources,
- Register - By instructions

i) Implied mode :- operand specified in inst. 8 or 16 bit data is part of inst. Zero AI.
Eg. CLC

ii) Immediate Mode :- Data placed in address field of inst one AI.
Eg: MOV AL, 35H
Limitation → Range of data is limited by size of address field.

iii) Register mode :- operand is placed in 8 or 16 bit register, it is registered specific mode.
Eg. MOV AX, CX

iv) Register indirect mode :- Data will be placed in memory location, address of memory will place in register. Register will be part of instruction.

Eg: MOV AX, [BX]

v) Auto index or increment mode :-
- Address of operand is placed in register specified in instruction.
- After accessing operand, address in register is incremented to point to next memory location
Eg. ADD R1, (R2)+

3. vi) Direct addressing :- operands address is given in inst an 8 bit or 16 bit element.

eg: ADD AL, [0301]

vii) Indirect addressing :- Address field contains the address of EA, requires 2 references a. Register indirect b. Memory indirect.

viii) Indexed Addressing mode :- Operands address is sum of content in index register SI or DI and 8 bit or 16 bit displacement.

Eg: MOV AX [SI+05]

ix) Based addressing mode :- EA is obtained by adding base register value to indexed register.

eg: ADD AX, [SI+BX]

3. Instruction set of 8086 microprocessor :-

- Data transfer :-

1) MOV :- Copy the byte or word from source to distination
2) PPUSH :- Used to put word at top of stack.
3) POP :- Used to get word from top of stack to the provided location.
4) POPA :- used to get works from stack to all registers.
5) XCHG :- Used to exchange the data from two location
6) XLAT :- Used to traskate byte in AL using table.
7) IN :- Used to read byte or word from provided register from memory.
8) Out :- used to send out byte or word from Ax.
9) LEA :- used to load the address of operand into provided register.
10) LDS :- Used to load DS register and other provided register from memory.
11) LES :- used to load LES register & other.
12) LAHF :- Used to store AH register to low byte of flag.
13) SAHF :- used to load AH register to flag.
14) PUSHF :- Copy of flag register at top of stack.
15) POPF :- Copy word at top of stack.

- Arithmetic instrictions :-

1) ADD - Used to add given byte or word.
2) ADC - used add with carry.
3) INC - increment provided by 1.
4) AAA - Adjust ASCII after addition
5) DAA - Adjust decimal after add/sub op.
6) SUB - Sub byte from byte/word.

3. 7) SBB - used to do substraction with borrow.
8) DEC - Decrement by 1.
9) NPG - negate by each bit 1's 2's complement.
10) CMP - compare 2 provided byte/ word.
11) DAS - Adjust decimal after Sub.
12) AAS - Adjust ASCII Code after Sub.
13) MUL - multiply unsigned byte by byte/word by word.
14) IMUL - multiply signed byte by byte/ word by word.
15) AAM - Adjust ASSCII code after Mul.
16) DIV - Divide unsigned word by byte.
17) AAD - Adjust ASCII code after div.
18) IDIV - Divide signed word by byte
19) CBW - Fill upper byte of word with copies of sign bit of lower byte
20) CWD - Fill upper word of double word.

- Bit manipulation instructions :-

1) NOT - Invert each bit of byte or word.
2) AND - adding each bit in byte/ word.
3) OR - used to multiplying each bit in byte/word.
4) XOR - Perfor Ex-OR Operation over each bit.
5) Test - Add operands to update flags
6) SHL / SAL - shift bits of byte/words towards left. put zero in LSB.
7) SHR - shift bits of byte/words towards right. Put zero is MSB.
8) SAR - shift bits of byte towards right and copy ~~new~~ old MSB into new MSB.
9) ROL - rotate bits towards left & to CF.
10) ROR - rotate bits towards right & to CF.
11) RCR - rotate bits towards right LSB → CF → MSB.

3. 12) RCL - rotate bits towards left

String instructions:-

1) REP - Repeat inst. till CX ≠ 0.
2) REPE/ REPZ - repeat inst. till CX ≠ 0 / ZF = 1.
3) REPNE/REPNZ - repeat inst. till CX = 0 / ZF = 1.
4) MOVS/ B/W - Move one string to another.
5) COMS/COMSB - compare two string bytes / words.
6) NS/NSB/ NNSW - Input string from I/O part to memory
7) OUTS/OUTSB/OUTSW - output string from I/O part.
8) SCAS/ SCASB - Scan a string & compare its byte.
9) LODS/ LODSB - Store string byte into AL.

Program execution transfer instructions:-

1) CALL - call procedure & save their return address to register
2) RET - return from procesdure to main program
3) JMP - Jump to provided address
4) JA
5) JAE
6) JC - Jump if carry flag = 1
7) JE/JZ - Jump if ZF = 1
8) JG/JNLE 9) JGE/JNL 10) JL/ JNGE 11) JEE/JNG
12) JNC - Jump if no carry flag
13) JNE/JNZ - jump if ZF = 0
14) JNO - jump if over flow flag, OF = 0.
15) JPO - jump if parity flag, PF = 0
16) JNS - jump if not sign SF = 0
17) JO - jum if overflow flag, OF = 1

3. 18) JP / JPE - Jump if PF = 1
19) JS - Jump if Sign flag, SF = 1

Processor control instruction :-

1) STC - Set carry flag CF = 1
2) CLC = Clear / rest CF to 0.
3) CMC - Post complement at state of CF.
4) STD - set direction flag, DF = 1
5) CLD - Clear / rest DF to 0.
6) STI - set interrupt flag, IF = 1.
7) CLI - Clear / rest IF to 0.

Iteration control instruction :-

1) LOOP - Loop group of instruction until condition satisfies i.e CX = 0
2) LOOPE / LOOPZ
3) LOOPNE / LOOPNZ - Group of inst till satisfies ZF = 0 & CF = 0

Interrupt instruction :-

1) INT 2) INTO 3) IRET

2. **Assembler Directives.**

Ans:

1. DATA copy or Transfer instruction –

   a. MOV: This instruction copies a word or a byte of data from some source to a destination.
   b. PUSH: Push to Stack - This instruction pushes the contents of the specified register/memory location on to the stack.
   c. POP: Pop from Stack - This instruction when executed, loads the specified register/memory location with the contents of the memory location.
   d. XCHG: Exchange byte or word - This instruction exchanges the contents of the specified source and destination operands.
   e. XLAT: Translate byte using look-up table.

2. Input and output port transfer instruction –

   a. IN: Copy a byte or word from specified port to accumulator.
   b. OUT: Copy a byte or word from accumulator specified port.
   c. LEA: Load effective address of operand in specified register.
   d. LDS: Load DS register and other specified register from memory.
   e. LES: Load ES register and other specified register from memory.

3. Arithmetic Instructions –

   a. ADD: The add instruction adds the contents of the source operand to the destination operand.
   b. ADC: Add with Carry - This instruction performs the same operation as ADD instruction, but adds the carry flag to the result.
   c. SUB: Subtract - The subtract instruction subtracts the source operand from the destination operand and the result is left in the destination operand.
   d. SBB: Subtract with Borrow - The subtract with borrow instruction subtracts the source operand and the borrow flag (CF) which may reflect the result of the previous calculations, from the destination operand.
   e. INC: Increment - This instruction increases the contents of the specified Register or memory location by 1. Immediate data cannot be operand of this instruction.
   f. DEC: Decrement - The decrement instruction subtracts 1 from the contents of the specified register or memory location.
   g. MUL: Unsigned Multiplication Byte or Word - This instruction multiplies an unsigned byte or word by the contents of AL.

h. DIV: Unsigned division - This instruction is used to divide an unsigned word by a byte or to divide an unsigned double word by a word.

4. Logical instructions –
   a. Logical AND
   b. Logical OR
   c. Logical NOR
   d. Logical XOR
   e. TEST

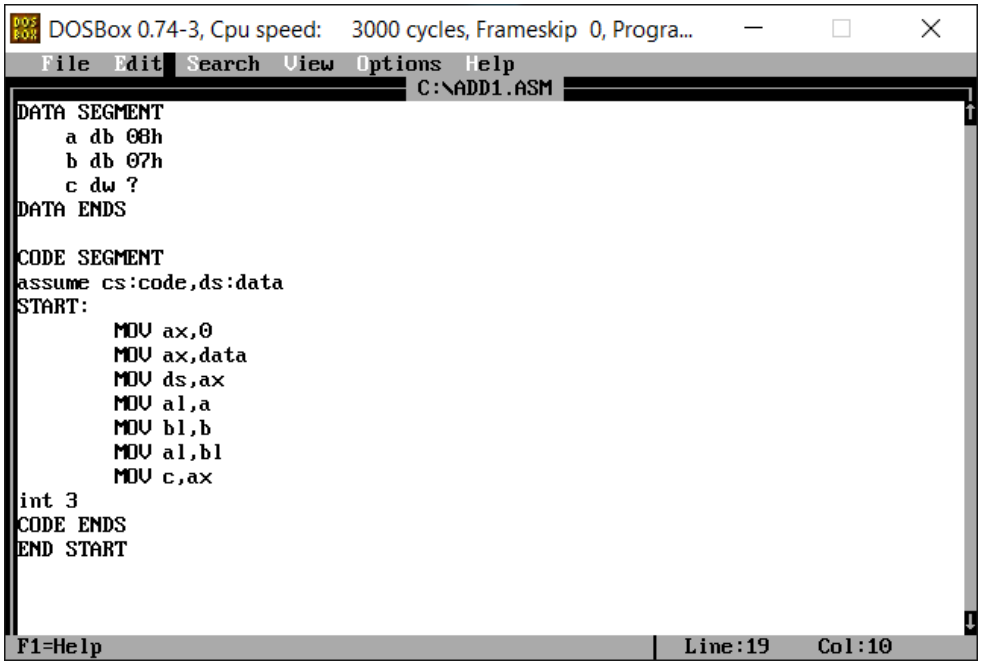
- Editor – Editor is used to edit particular program in window. It is very useful to edit such programs effectively. DOSBOX includes editor option inbuilt with edit command.
- Assembler – It converts low level computer language to machine level language in order to perform specific tasks.
- Debugger - It is computer program or function that used to test and debug specific program by validating the output of that program.
- Simulator - A device or program that enables the operator to reproduce or represent under test conditions phenomena likely to occur in actual performance.
- Emulator - It is hardware or software that enables one computer system to behave like another system mostly computer systems or hardware systems.
- Function calls - BIOS interrupt calls can be thought of as a mechanism for passing messages between BIOS and BIOS client software such as an operating system.

3. 8 Bit Arithmetic Programs –

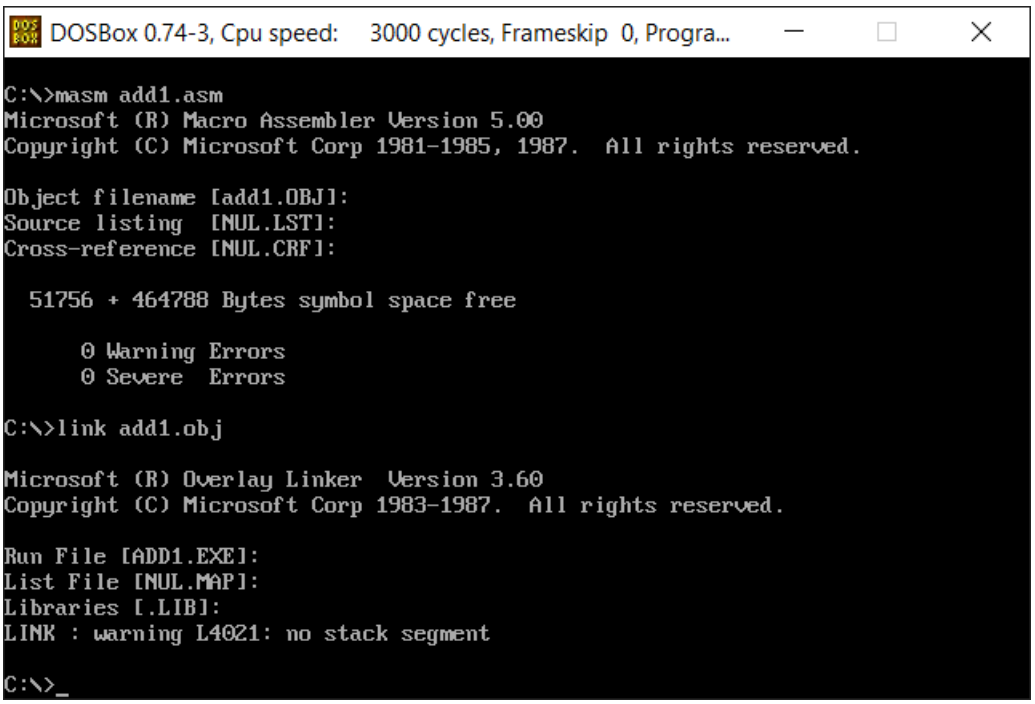
1. 8 Bit addition:

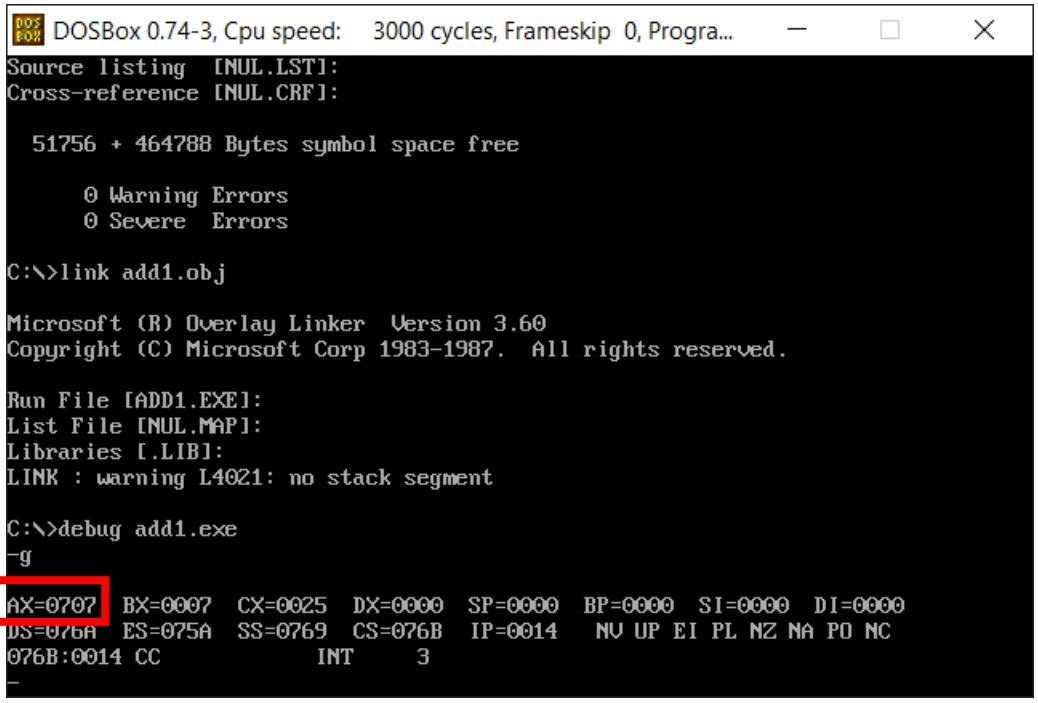
Program –

```
DATA SEGMENT
    a db 08h
    b db 07h
    c dw ?
DATA ENDS
CODE SEGMENT
assume cs:code,ds:data
START:
        MOV ax,0
    MOV ax,data
    MOV ds,ax
    MOV al,a
    MOV bl,b
    MOV al,bl
    MOV c,ax
int 3
CODE ENDS
END START
```



Output –

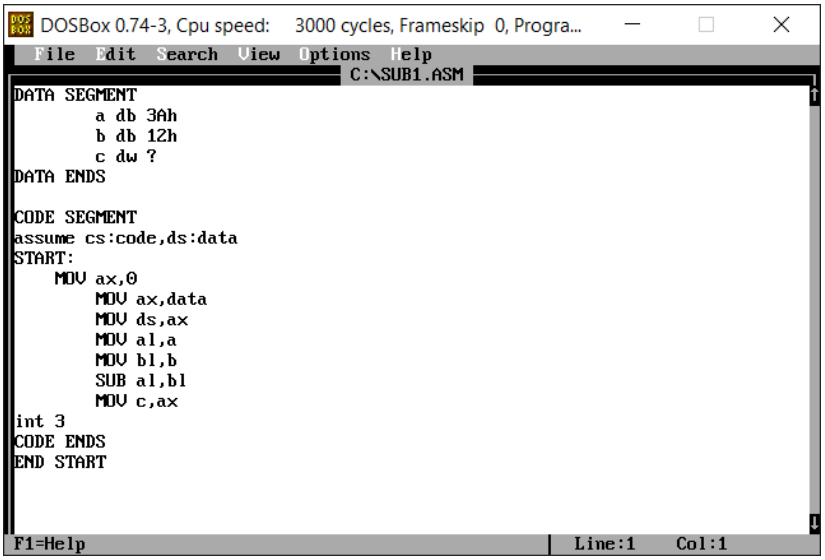

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>masm add1.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.
Object filename [add1.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
51756 + 464788 Bytes symbol space free
0 Warning Errors
0 Severe Errors
C:\>link add1.obj
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
Run File [ADD1.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
C:\>_


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
51756 + 464788 Bytes symbol space free
0 Warning Errors
0 Severe Errors
C:\>link add1.obj
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
Run File [ADD1.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
C:\>debug add1.exe
-g
AX=0707 BX=0007 CX=0025 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0014 NV UP EI PL NZ NA PO NC
076B:0014 CC INT 3
-

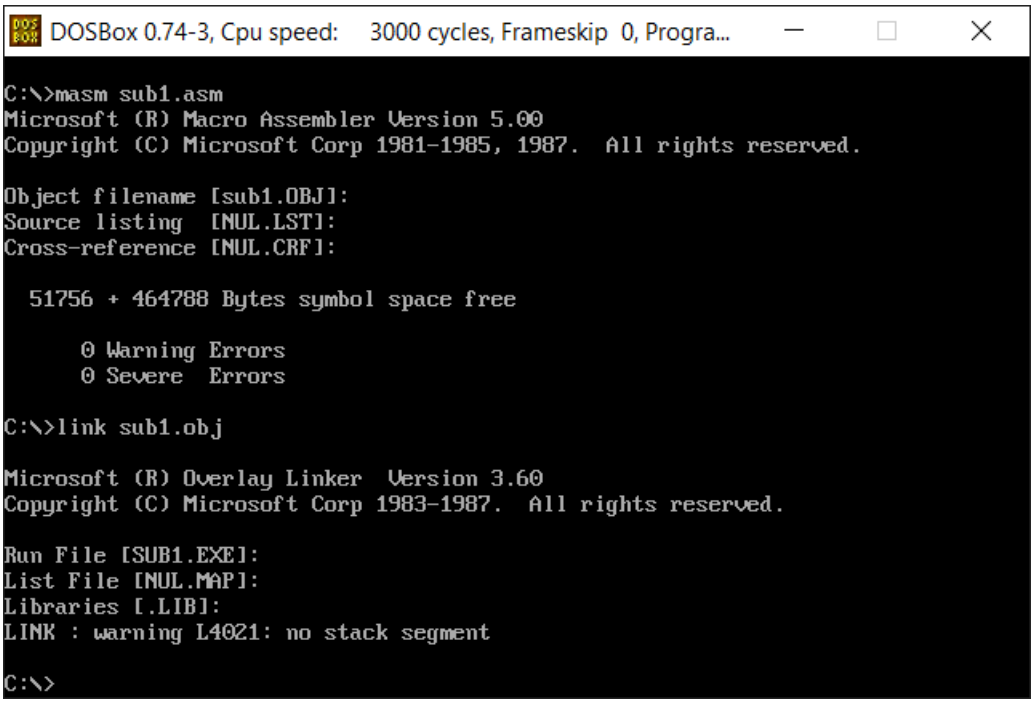
2. 8 Bit Subtraction:

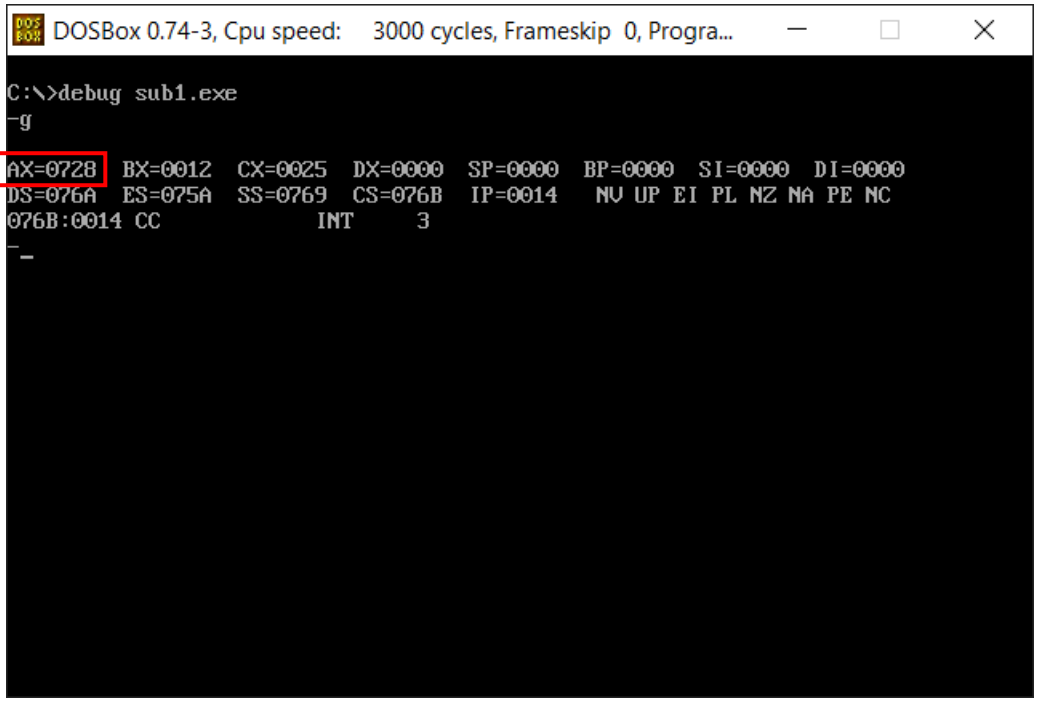
Program -

```
DATA SEGMENT
  a db 3Ah
  b db 12h
  c dw ?
DATA ENDS
CODE SEGMENT
assume cs:code,ds:data
START:
   MOV ax,0
  MOV ax,data
  MOV ds,ax
  MOV al,a
  MOV bl,b
  SUB al,bl
  MOV c,ax
int 3
CODE ENDS
END START
```



Output –

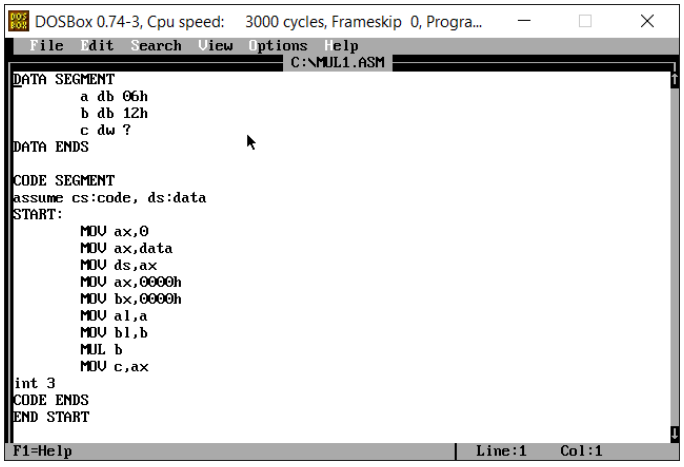

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>masm sub1.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.
Object filename [sub1.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
51756 + 464788 Bytes symbol space free
0 Warning Errors
0 Severe Errors
C:\>link sub1.obj
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
Run File [SUB1.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
C:\>


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>debug sub1.exe
-g
AX=0728 BX=0012 CX=0025 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0014 NV UP EI PL NZ NA PE NC
076B:0014 CC INT 3
-

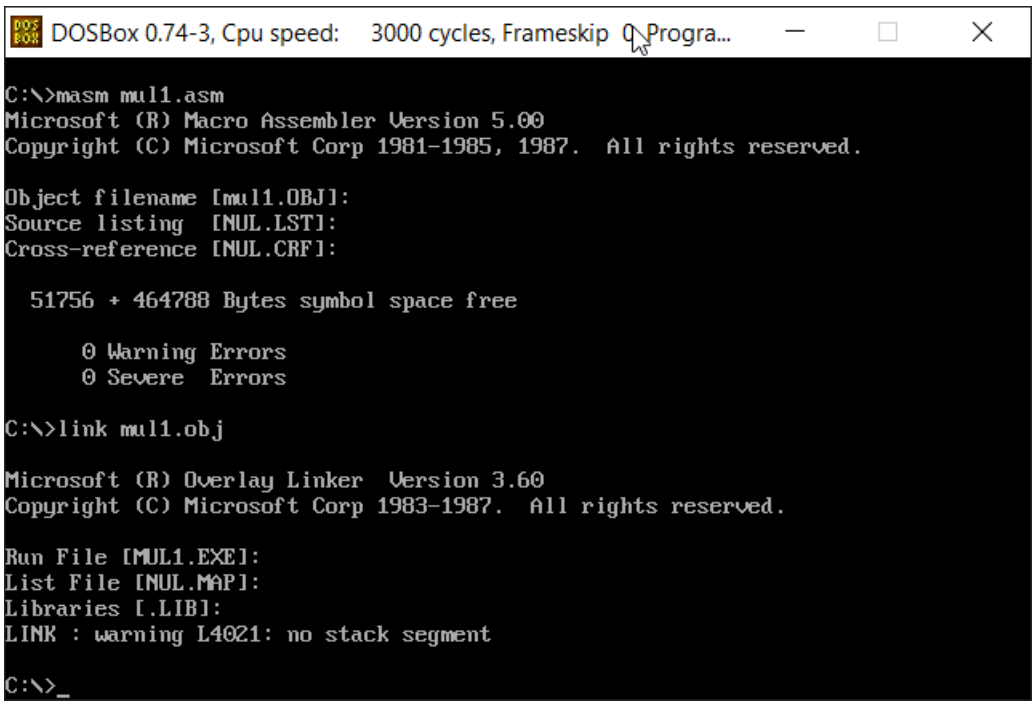
3. 8 Bit Multiplication:

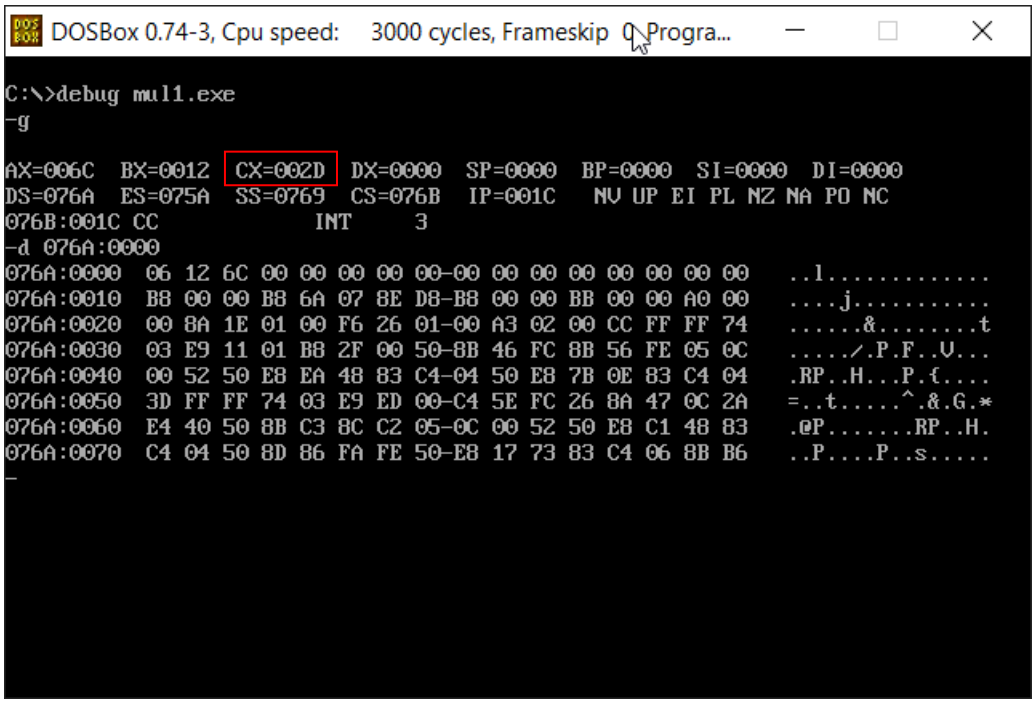
Program –

```
DATA SEGMENT
    a db 06h
    b db 12h
    c dw ?
DATA ENDS
CODE SEGMENT
assume cs:code, ds:data
START:
    MOV ax,0
    MOV ax,data
    MOV ds,ax
    MOV ax,0000h
    MOV bx,0000h
    MOV al,a
    MOV bl,b
    MUL b
    MOV c,ax
int 3
CODE ENDS
END START
```



Output –


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>masm mul1.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.
Object filename [mul1.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
51756 + 464788 Bytes symbol space free
0 Warning Errors
0 Severe Errors
C:\>link mul1.obj
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
Run File [MUL1.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
C:\>_


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>debug mul1.exe
-g
AX=006C BX=0012 CX=002D DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=001C NV UP EI PL NZ NA PO NC
076B:001C CC INT 3
-d 076A:0000
076A:0000 06 12 6C 00 00 00 00 00-00 00 00 00 00 00 00 00 ..l.....
076A:0010 B8 00 00 B8 6A 07 8E D8-B8 00 00 BB 00 00 A0 00 ....j.....
076A:0020 00 8A 1E 01 00 F6 26 01-00 A3 02 00 CC FF FF 74 ......&.....t
076A:0030 03 E9 11 01 B8 2F 00 50-8B 46 FC 8B 56 FE 05 0C ...../.P.F..V...
076A:0040 00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04 .RP..H...P.{....
076A:0050 3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A =..t.....^.&.G.*
076A:0060 E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83 .@P.....RP..H.
076A:0070 C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6 ..P....P..s.....
-

4. 8 Bit Division:

Program –

```
DATA SEGMENT
  a db 34h
  b db 02h
  c dw ?
DATA ENDS
CODE SEGMENT
assume cs:code, ds:data
START:
  MOV ax,0
  MOV ax,data
  MOV ds,ax
  MOV ax,0000h
  MOV bx,0000h
  MOV al,a
  MOV bl,b
  DIV b
  MOV c,ax
int 3
CODE ENDS
END START
```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

File Edit Search View Options Help

C:\DIV1.ASM

```
DATA SEGMENT
        a db 34h
        b db 02h
        c dw ?
DATA ENDS

CODE SEGMENT
assume cs:code, ds:data
START:
        MOV ax,0
        MOV ax,data
        MOV ds,ax
        MOV ax,0000h
        MOV bx,0000h
        MOV al,a
        MOV bl,b
        DIV b
        MOV c,ax
int 3
CODE ENDS
END START
```

F1=Help | Line:1 Col:1

Output –

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
C:\>masm div1.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [div1.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51756 + 464788 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link div1.obj

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

Run File [DIV1.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>_
```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

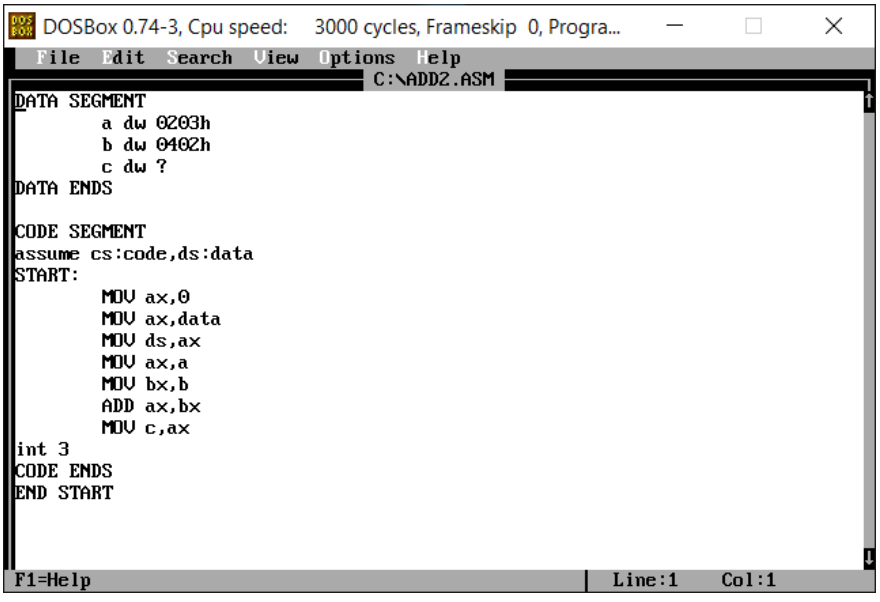
```
C:\>debug div1.exe
-g

AX=001A  BX=0002  CX=002D  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=001C   NV UP EI PL NZ NA PO NC
076B:001C CC            INT     3
-d 076A:0000
076A:0000  34 02 1A 00 00 00 00 00-00 00 00 00 00 00 00 00   4...............
076A:0010  B8 00 00 B8 6A 07 8E D8-B8 00 00 BB 00 00 A0 00   ....j...........
076A:0020  00 8A 1E 01 00 F6 36 01-00 A3 02 00 CC FF FF 74   ......6........t
076A:0030  03 E9 11 01 B8 2F 00 50-8B 46 FC 8B 56 FE 05 0C   ...../.P.F..V...
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04   .RP..H...P.{....
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A   =..t.....^.&.G.*
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83   .@P.......RP..H.
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6   ..P....P..s.....
-_
```

4. 16 Bit Arithmetic Programs –

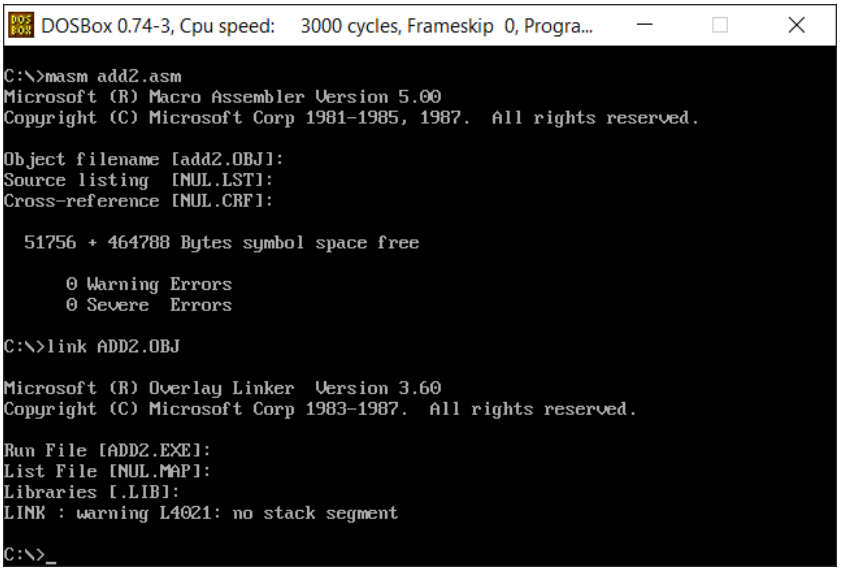
1. 16 Bit addition:

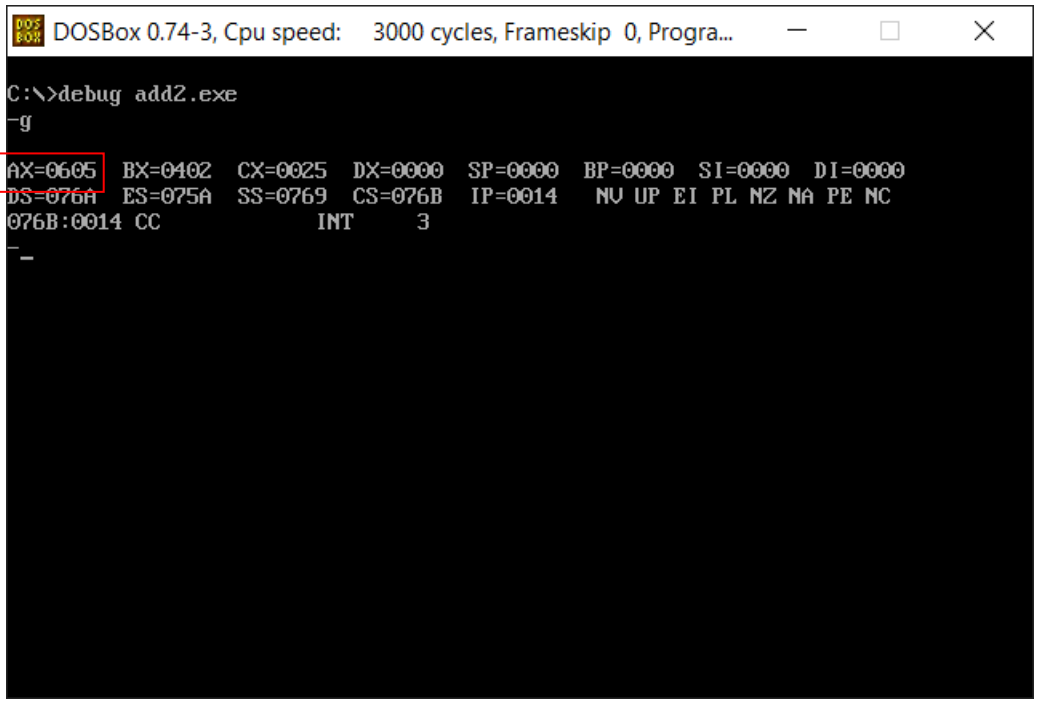
Program –

```
DATA SEGMENT
    a dw 0203h
    b dw 0402h
    c dw ?
DATA ENDS
CODE SEGMENT
assume cs:code,ds:data
START:
    MOV ax,0
    MOV ax,data
    MOV ds,ax
    MOV ax,a
    MOV bx,b
    ADD ax,bx
    MOV c,ax
int 3
CODE ENDS
END START
```



Output –

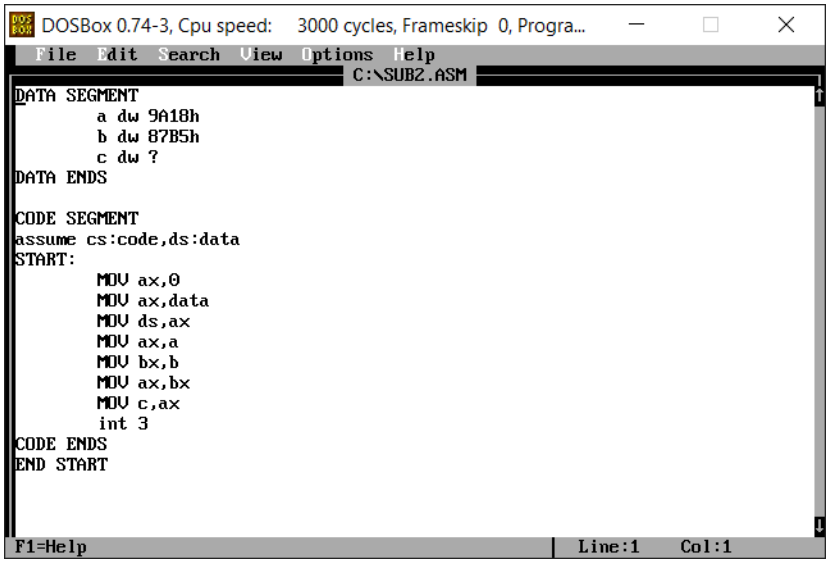

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>masm add2.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.
Object filename [add2.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
51756 + 464788 Bytes symbol space free
0 Warning Errors
0 Severe Errors
C:\>link ADD2.OBJ
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
Run File [ADD2.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
C:\>_


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>debug add2.exe
-g
AX=0605 BX=0402 CX=0025 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0014 NV UP EI PL NZ NA PE NC
076B:0014 CC INT 3
-_

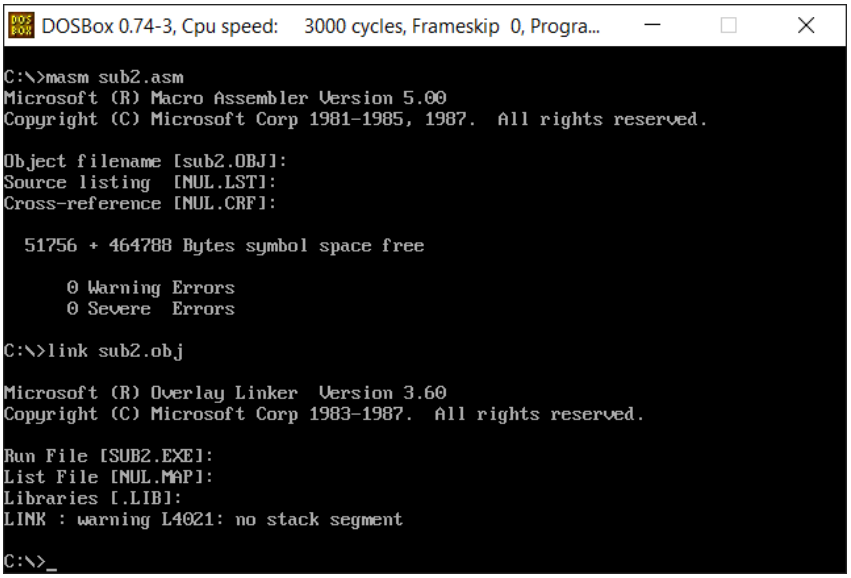
2. 16 Bit subtraction:

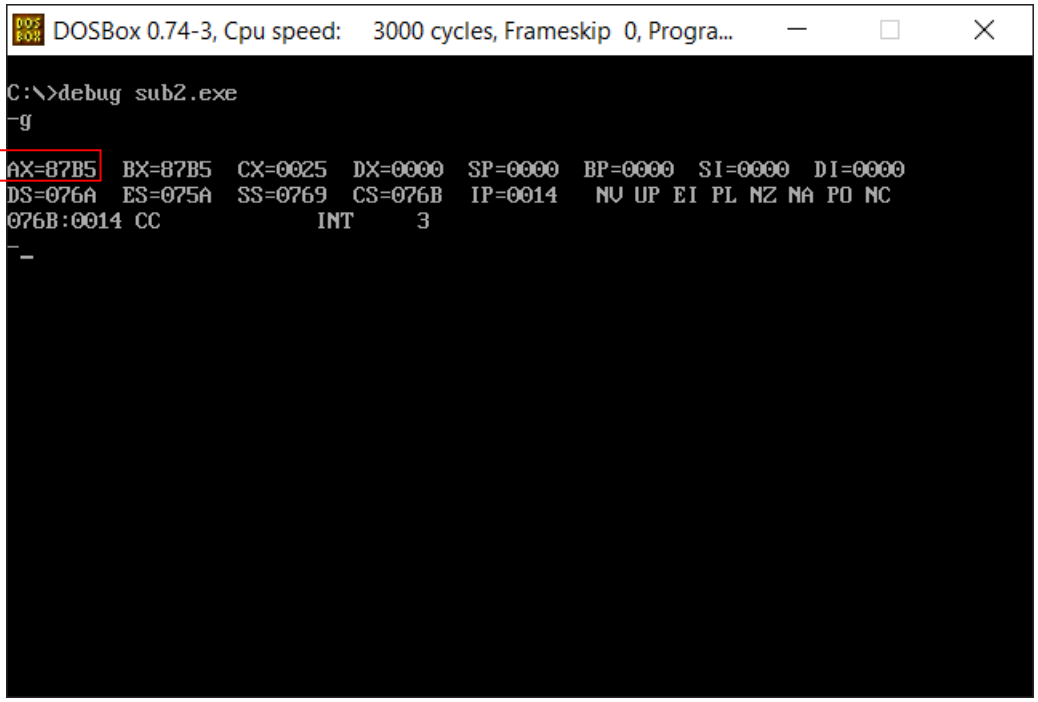
Program -

```
DATA SEGMENT
 a dw 9A18h
 b dw 87B5h
 c dw ?
DATA ENDS
CODE SEGMENT
assume cs:code,ds:data
START:
 MOV ax,0
 MOV ax,data
 MOV ds,ax
 MOV ax,a
 MOV bx,b
 MOV ax,bx
 MOV c,ax
 int 3
CODE ENDS
END START
```



Output –


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>masm sub2.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987. All rights reserved.
Object filename [sub2.OBJ]:
Source listing [NUL.LST]:
Cross-reference [NUL.CRF]:
51756 + 464788 Bytes symbol space free
0 Warning Errors
0 Severe Errors
C:\>link sub2.obj
Microsoft (R) Overlay Linker Version 3.60
Copyright (C) Microsoft Corp 1983-1987. All rights reserved.
Run File [SUB2.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment
C:\>_


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...
C:\>debug sub2.exe
-g
AX=87B5 BX=87B5 CX=0025 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=076A ES=075A SS=0769 CS=076B IP=0014 NV UP EI PL NZ NA PO NC
076B:0014 CC INT 3
-_

3. 16 Bit multiplication:

Program –

```
DATA SEGMENT
    a dw 2341h
    b dw 6687h
    c dd ?
DATA ENDS
CODE SEGMENT
assume ds:data, cs:code
START:
    MOV ax,0
    MOV ax,data
    MOV ds,ax
    MOV ax,a
    MOV bx,b
    MUL bx
    MOV word ptr c,ax
    MOV word ptr c+2,dx
    int 3
CODE ENDS
END START
```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

File Edit Search View Options Help

C:\MUL2.ASM

```
DATA SEGMENT
        a dw 2341h
        b dw 6687h
        c dd ?
DATA ENDS

CODE SEGMENT
assume ds:data, cs:code
START:
        MOV ax,0
        MOV ax,data
        MOV ds,ax
        MOV ax,a
        MOV bx,b
        MUL bx
        MOV word ptr c,ax
        MOV word ptr c+2,dx
        int 3
CODE ENDS
END START
```

F1=Help | Line:1 Col:1

Output –

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
C:\>masm mul2.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [mul2.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51680 + 464864 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link mul2.obj

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

Run File [MUL2.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>_
```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Progra...

```
C:\>debug mul2.exe
-g

AX=7D47  BX=6687  CX=0029  DX=0E1E  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=0018   OV UP EI PL NZ NA PO CY
076B:0018 CC            INT     3
-d 076A:0000
076A:0000  41 23 87 66 47 7D 1E 0E-00 00 00 00 00 00 00 00   A#.fG}.........
076A:0010  B8 00 00 B8 6A 07 8E D8-A1 00 00 8B 1E 02 00 F7   ....j...........
076A:0020  E3 A3 04 00 89 16 06 00-CC 83 C4 04 3D FF FF 74   ............=..t
076A:0030  03 E9 11 01 B8 2F 00 50-8B 46 FC 8B 56 FE 05 0C   ...../.P.F..V...
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04   .RP..H...P.{....
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A   =..t.....^.&.G.*
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83   .@P.......RP..H.
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6   ..P....P..s.....
-
```

4. 16 Bit Division:

Program –

```
DATA SEGMENT
  a dw 8324h
  b dw 0002h
  c dw ?
DATA ENDS

CODE SEGMENT
assume ds:data, cs:code
START:
  MOV ax,0
  MOV ax,data
  MOV ds,ax
  MOV ax,a
  MOV bx,b
  DIV bx
  MOV c,ax
  int 3
CODE ENDS
END START
```

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
File  Edit  Search  View  Options  Help
                        C:\DIV2.ASM
DATA SEGMENT
        a dw 8324h
        b dw 0002h
        c dw ?
DATA ENDS

CODE SEGMENT
assume ds:data, cs:code
START:
        MOV ax,0
        MOV ax,data
        MOV ds,ax
        MOV ax,a
        MOV bx,b
        DIV bx
        MOV c,ax
        int 3
CODE ENDS
END START


F1=Help                                          Line:1     Col:1
```

Output –

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...

C:\>masm div2.asm
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Object filename [div2.OBJ]:
Source listing  [NUL.LST]:
Cross-reference [NUL.CRF]:

  51756 + 464788 Bytes symbol space free

      0 Warning Errors
      0 Severe  Errors

C:\>link div2.obj

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

Run File [DIV2.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:
LINK : warning L4021: no stack segment

C:\>
```

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...

C:\>debug div2.exe
-g

AX=4192  BX=0002  CX=0025  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076A  ES=075A  SS=0769  CS=076B  IP=0014   NV UP EI PL NZ NA PO NC
076B:0014 CC            INT     3
-d 076A:0000
076A:0000  24 83 02 00 92 41 00 00-00 00 00 00 00 00 00 00   $....A..........
076A:0010  B8 00 00 B8 6A 07 8E D8-A1 00 00 8B 1E 02 00 F7   ....j...........
076A:0020  F3 A3 04 00 CC 50 E8 9F-0E 83 C4 04 3D FF FF 74   .....P......=..t
076A:0030  03 E9 11 01 B8 2F 00 50-8B 46 FC 8B 56 FE 05 0C   ...../.P.F..V...
076A:0040  00 52 50 E8 EA 48 83 C4-04 50 E8 7B 0E 83 C4 04   .RP..H...P.{....
076A:0050  3D FF FF 74 03 E9 ED 00-C4 5E FC 26 8A 47 0C 2A   =..t.....^.&.G.*
076A:0060  E4 40 50 8B C3 8C C2 05-0C 00 52 50 E8 C1 48 83   .@P.......RP..H.
076A:0070  C4 04 50 8D 86 FA FE 50-E8 17 73 83 C4 06 8B B6   ..P....P..s.....
-_
```