DIGITAL ASSIGNMENT : GENERAL INSTRUCTIONS

MICROPROCESSOR AND INTERFACING(D1)[MRS SHOBHA REKH]

SEPTEMBER 13, 2022

ANIRUDH VADERA
20BCE2940

REGISTRATION NUMBER: 20BCE2940

QUESTION TO BE CHOSEN: 2 + 0 + 2 + 9 + 4 + 0 = 17 = 1 + 7 = 8

QUESTION 8 Part A:

Find if the given number is a prime number.

Algorithm:

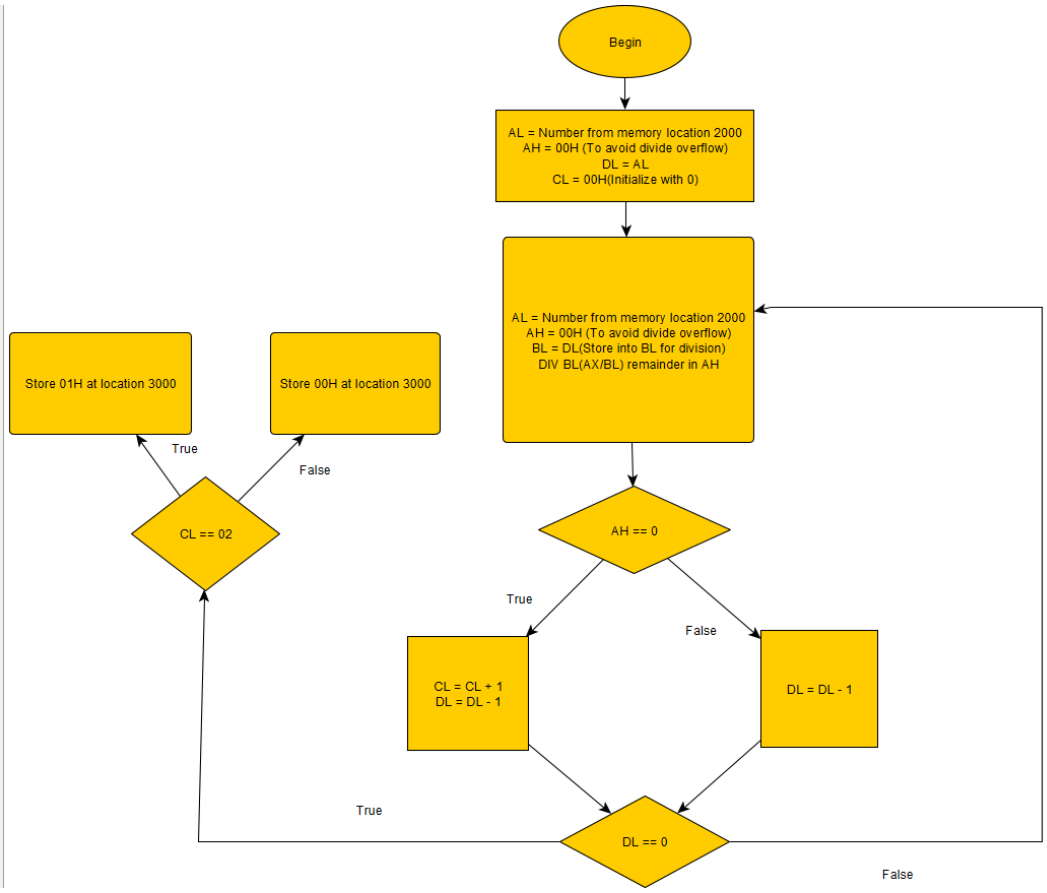
1. Take n as input
2. Run a loop from i = n to 1. For each iteration, check if i divides n completely or not. If it does, then i is n's divisor
3. Keep a count of the total number of divisors of n
4. If the count of divisors is 2, then the number is prime, else composite

In Assembly Language:

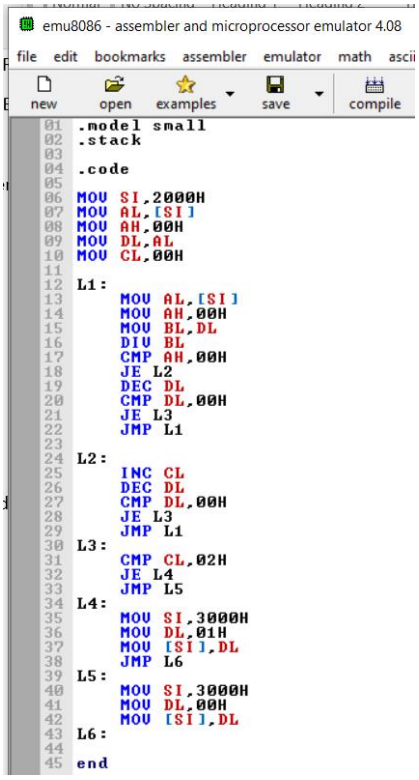
DL goes from n to 1 each time we divide the given number by DL(Actually the given number is in AL then AH is 00H(to avoid divide overflow) and we move DL to BL then execute DIV BL)

1. Load the data from memory from location 2000 into AL register
2. Move 00H into AH in order to avoid Divide overflow error
3. Move AL value into DL register which will be decremented later
4. Move 00H into CL register it stores the number of divisors of the given number at the end if the value of CL is 02H then the given number is prime else its not a prime number
5. We start with label L1:
6. We take the number into AL and move 00H into AH
7. Then move DL into BL in order to perform division operation
8. Compare the value of AH(AH Stores the remainder of division) with 00H after division
9. Use JE(Jump on equal) after CMP to label2 now if the number has a divisor then the value of CL will be incremented
10. In L2 we just increment the value of CL
11. We then decrease DL by 1
12. If DL is 00H then we jump to L3 otherwise we loop back again in L1
13. In L3 we simply compare CL by 02H if it is 02H then we jump to L4 otherwise to L5
14. In L4 it means the number of divisors of the given number is only 2 hence a number is prime hence we store the value 01H at location 3000H
15. In L5 it means the number of divisors of the given number is more than 2 hence a number is not prime hence we store the value 00H at location 3000H
16. We end the program now

FlowChart:



Code Screenshot:




```
.model small
.stack

.code

MOV SI,2000H
MOV AL,[SI]
MOV AH,00H
MOV DL,AL
MOV CL,00H

L1:
    MOV AL,[SI]
    MOV AH,00H
    MOV BL,DL
    DIV BL
    CMP AH,00H
    JE L2
    DEC DL
    CMP DL,00H
    JE L3
    JMP L1

L2:
    INC CL
    DEC DL
    CMP DL,00H
    JE L3
    JMP L1
L3:
    CMP CL,02H
    JE L4
    JMP L5
L4:
    MOV SI,3000H
    MOV DL,01H
    MOV [SI],DL
    JMP L6
L5:
    MOV SI,3000H
    MOV DL,00H
    MOV [SI],DL
L6:

end
```

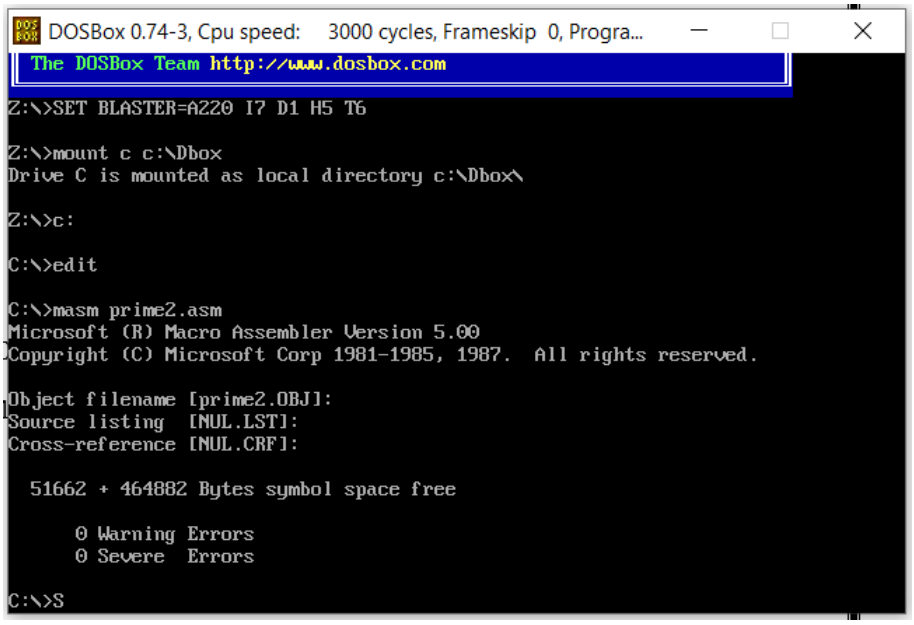
Assembly Language Code:

```
.model small
.stack

.code

MOV SI,2000H
MOV AL,[SI]
MOV AH,00H
MOV DL,AL
MOV CL,00H

L1:
   MOV AL,[SI]
   MOV AH,00H
   MOV BL,DL
   DIV BL
   CMP AH,00H
   JE L2
   DEC DL
   CMP DL,00H
   JE L3
   JMP L1

L2:
   INC CL
   DEC DL
   CMP DL,00H
   JE L3
```

```
    JMP L1
L3:
    CMP CL,02H
    JE L4
    JMP L5
L4:
    MOV SI,3000H
    MOV DL,01H
    MOV [SI],DL
    JMP L6
L5:
    MOV SI,3000H
    MOV DL,00H
    MOV [SI],DL
L6:

end
```

Execution Proof:

Command: masm prime2.asm



Command: link prime2.obj

```
C:\>link prime2.obj

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

Run File [PRIME2.EXE]:
List File [NUL.MAP]:
Libraries [.LIB]:

C:\>S
```

Command: debug prime2.exe

```
C:\>debug prime2.exe
-S
```

Given Input at location 2000H

Output at Location 3000H

If value stored at location 3000H is 00H – The Given Number is not prime

If value stored at location 3000H is 01H – The Given Number is prime

Giving Inputs:

Given Number : **03** (Expected Output **01H** at 3000H)

```
-e ds:2000
075A:2000  7F.03

-S
```

A screenshot of process in middle:

```
DOSBox 0.74-3, Cpu speed:   3000 cycles, Frameskip  0, Progra...
AX=FFFF  BX=0000  CX=0045  DX=0000  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0003   NV UP EI PL NZ NA PO NC
076A:0003 8A04          MOV     AL,[SI]                          DS:2000=03
-t

AX=FF03  BX=0000  CX=0045  DX=0000  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0005   NV UP EI PL NZ NA PO NC
076A:0005 B400          MOV     AH,00
-t

AX=0003  BX=0000  CX=0045  DX=0000  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0007   NV UP EI PL NZ NA PO NC
076A:0007 8AD0          MOV     DL,AL
-t

AX=0003  BX=0000  CX=0045  DX=0003  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0009   NV UP EI PL NZ NA PO NC
076A:0009 B100          MOV     CL,00
-t

AX=0003  BX=0000  CX=0000  DX=0003  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=000B   NV UP EI PL NZ NA PO NC
076A:000B 8A04          MOV     AL,[SI]                          DS:2000=03
-S
```

The Output is:

```
-d ds:3000
075A:3000  01 06 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   ...P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S
```

```
-d ds:3000
075A:3000  01
```

Hence **01H** is stored at **location 3000** hence **03 is a prime number**

Giving Inputs:

Given Number : **12(0C)** (Expected Output **00H at 3000H**)

```
C:\>debug prime2.exe
-e ds:2000
075A:2000  03.0C
```

A screenshot of process in middle:

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

AX=0001  BX=000C  CX=0001  DX=000B  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0028   NV UP EI PL NZ NA PO NC
076A:0028 7402          JZ      002C
-t

AX=0001  BX=000C  CX=0001  DX=000B  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=002A   NV UP EI PL NZ NA PO NC
076A:002A EBDF          JMP     000B
-t

AX=0001  BX=000C  CX=0001  DX=000B  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=000B   NV UP EI PL NZ NA PO NC
076A:000B 8A04          MOV     AL,[SI]                          DS:2000=0C
-t

AX=000C  BX=000C  CX=0001  DX=000B  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=000D   NV UP EI PL NZ NA PO NC
076A:000D B400          MOV     AH,00
-t

AX=000C  BX=000C  CX=0001  DX=000B  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=000F   NV UP EI PL NZ NA PO NC
076A:000F 8ADA          MOV     BL,DL
-S
```

The Output is:

```
-d ds:3000
075A:3000  00 06 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   ...P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S_
```

```
-d ds:3000
075A:3000  00
```

Hence **00H** is stored at **location 3000** hence **12(0C) is not a prime number**

Giving Inputs:

Given Number : **05** (Expected **Output 01H at 3000H**)

```
C:\>debug prime2.exe
-e ds:2000
075A:2000  0C.05
```

A screenshot of process in middle:

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

AX=0001  BX=0005  CX=0000  DX=0005  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0013   NV UP EI PL NZ NA PO NC
076A:0013 80FC00        CMP     AH,00
-t

AX=0001  BX=0005  CX=0000  DX=0005  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0016   NV UP EI PL ZR NA PE NC
076A:0016 7409          JZ      0021
-t

AX=0001  BX=0005  CX=0000  DX=0005  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0021   NV UP EI PL ZR NA PE NC
076A:0021 FEC1          INC     CL
-t

AX=0001  BX=0005  CX=0001  DX=0005  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0023   NV UP EI PL NZ NA PO NC
076A:0023 FECA          DEC     DL
-t

AX=0001  BX=0005  CX=0001  DX=0004  SP=0400  BP=0000  SI=2000  DI=0000
DS=075A  ES=075A  SS=076F  CS=076A  IP=0025   NV UP EI PL NZ NA PO NC
076A:0025 80FA00        CMP     DL,00
-S_
```

The Output is:

```
-d ds:3000
075A:3000  01 06 00 50 8B 5E FE D1-E3 D1 E3 8B 36 7E 21 FF   ...P.^......6~!.
075A:3010  70 02 FF 30 E8 19 E4 83-C4 08 89 46 FA 89 56 FC   p..0.......F..V.
075A:3020  C4 5E FA 26 8A 47 05 BE-0A 27 8A 1C FF 04 2A FF   .^.&.G...'....*.
075A:3030  8B 36 E0 25 88 00 A1 26-22 48 50 E8 3A 13 83 C4   .6.%...&"HP.:...
075A:3040  02 5E 8B E5 5D C3 55 8B-EC 83 EC 06 A1 A8 09 05   .^..].U.........
075A:3050  06 00 3B 06 A8 09 73 08-8B 46 04 8B E5 5D C3 90   ..;...s..F...]..
075A:3060  8B 46 04 89 46 FA 8B 46-06 8B 56 08 89 46 FC 89   .F..F..F..V..F..
075A:3070  56 FE B8 FF FF 50 A1 A8-09 2B D2 03 C2 81 D2 94   V....P...+......
-S
```

```
-d ds:3000
075A:3000  01
```

Hence **01H** is stored at **location 3000** hence **05 is a prime number**

QUESTION 8 Part B:

Find the name of the processor available in your laptop. Identify the size of address and data.

Solution:

Name of the Processor: AMD Renoir (Ryzen 4800 APU)

Device specifications

Device name	LAPTOP-526LOF85
Processor	AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz

CodeName(Micro Architecture Used): Renoir (Zen 2)

Short Description:

Ryzen 7 4800H is a 64-bit octa-core high-end performance x86 mobile microprocessor introduced by AMD in early 2020. Fabricated on TSMC's 7-nanometer process and based on AMD's Zen 2 microarchitecture, the 4800H operates at a base frequency of 2.9 GHz with a TDP of 45 W and a boost frequency of up to 4.2 GHz.

General Specs	
Family	Ryzen 7
Series	4000
Locked	Yes
Frequency	2,900 MHz
Turbo Frequency	4,200 MHz
Clock multiplier	29

Microarchitecture	
ISA	x86-64 (x86)
Microarchitecture	Zen 2
Core Name	Renoir
Core Family	23
Core Model	96
Core Stepping	A1
Process	7 nm
Transistors	9,800,000,000
Technology	CMOS
Die	156 mm²
Word Size	64 bit
Cores	8
Threads	16
Max Memory	64 GiB

Address and Data Bus(Size Space) :

Given ISA : x86-64 (x86)

There are not yet any x86-64 systems that have a 64 bit address bus.

In order to find address bus size type use the following command:

grep 'address sizes' /proc/cpuinfo

```
MINGW64:/c/Users/Anirudh

Anirudh@LAPTOP-526LOF85 MINGW64 /c/Users/Anirudh
$ grep 'address sizes' /proc/cpuinfo
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual
address sizes   : 48 bits physical, 48 bits virtual

Anirudh@LAPTOP-526LOF85 MINGW64 /c/Users/Anirudh
$
```

We get to know that the address bus is of 48 bits

- ➔ **At most 2^{48} of bytes can be addressed for ROM, RAM, VRAM, IO ….**
- ➔ **At most 2^{48} of bytes of virtual memory can be addressed, per process.**

Space = 2^{48}B = 2^{8} TB = 256TB

Data Bus(Space):

Given ISA : x86-64 (x86)

The logical width of the data-bus, of an x86-64 is simply **64 bits(8-byte wide data bus).**

However, the physical size is whatever the manufacturer chooses. They can choose whatever they want, without affecting behaviour Eg. Multiplex 64 bits over 32 or 16 bits, or send two 64 bits over 32 or 16 bits, or send two 64 bit values over a 128-bit bus.

Maximum Possible (Data)Memory Spaces available: 2^64 = **1.8446744e+19 (18.4 exabytes) But in reality its way less than that.**

Space = 2^{64}B = 2^{4} Exabyte = 16EB