

**PRAKTIKUM SISTEM OPERASI**  
**MODUL 1**  
**PENGENALAN SISTEM PENGEMBANGAN OS**  
**DENGAN PC SIMULATOR “BOCHS”**

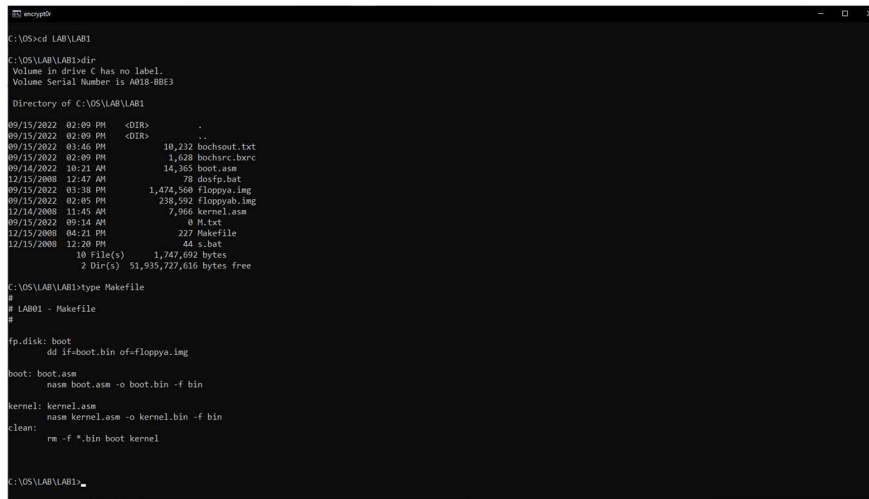


**Disusun Oleh :**  
**MUHAMMAD WAHYU SYAFI'UDDIN**  
**L200210056**

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**FAKULTAS KOMUNIKASI DAN INFORMATIKA**  
**UNIVERSITAS MUHAMMADIYAH SURAKARTA**  
**TAHUN 2021/2022**

## A. Langkah Kerja

1. Masuk ke direktori LAB/LAB1 dan melihat isi dari Makefile



```
C:\OS>cd LAB\LAB1

C:\OS\LAB\LAB1>dir
Volume in drive C has no label.
Volume Serial Number is A018-BBE3

Directory of C:\OS\LAB\LAB1

09/15/2022 02:09 PM <DIR>      .
09/15/2022 02:09 PM <DIR>      ..
09/15/2022 03:46 PM           10,232 bochsout.txt
09/15/2022 02:09 PM           1,628 bochsrc.barc
09/14/2022 10:21 AM           14,365 boot.asm
12/15/2008 12:47 AM             78 dosfp.bat
09/15/2022 03:38 PM       1,474,560 floppyb.img
09/15/2022 02:05 PM       230,592 floppyb.img
12/14/2008 11:45 AM           7,560 kernel.asm
09/15/2022 09:14 AM              0 H.txt
12/15/2008 04:21 PM           227 Makefile
12/15/2008 12:20 PM             44 *.bat
               10 File(s)       1,747,692 bytes
               2 Dir(s)       51,935,727,016 bytes free

C:\OS\LAB\LAB1>type Makefile
#
# LAB01 - Makefile
#

fp.disk: boot
dd if=boot.bin of=floppyb.img

boot: boot.asm
nasm boot.asm -o boot.bin -f bin

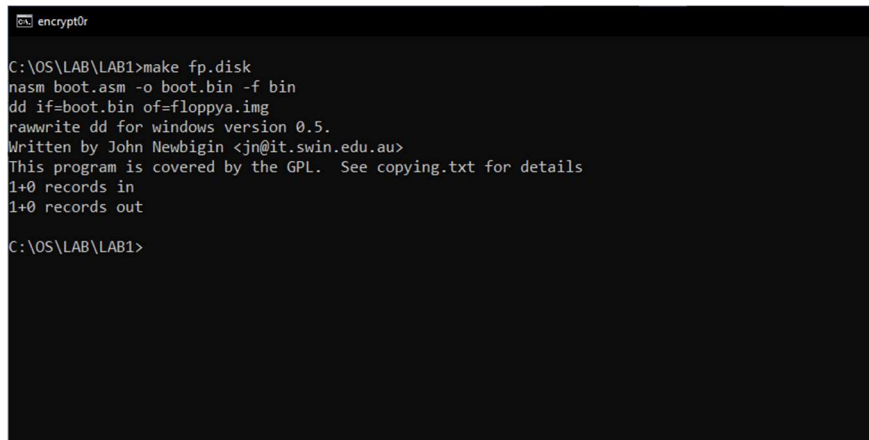
kernel: kernel.asm
nasm kernel.asm -o kernel.bin -f bin

clean:
rm -f *.bin boot kernel

C:\OS\LAB\LAB1>
```

Gambar 1.3 Menyunting file ‘Makefile’

2. Make fp.disk

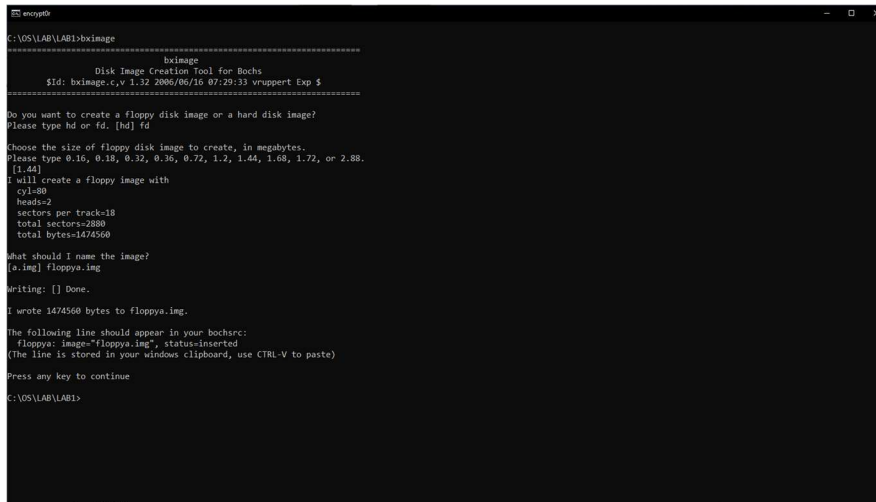


```
C:\OS\LAB\LAB1>make fp.disk
nasm boot.asm -o boot.bin -f bin
dd if=boot.bin of=floppyb.img
rawwrite dd for windows version 0.5.
Written by John Newbigin <jn@it.swin.edu.au>
This program is covered by the GPL. See copying.txt for details
1+0 records in
1+0 records out

C:\OS\LAB\LAB1>
```

Gambar 1.4 Menggunakan perintah Make

### 3. Membuat Floppya.img dengan bximage



```
C:\OS\LAB\LAB1>bximage

bximage
Disk Image Creation Tool for Bochs
$Id: bximage.c,v 1.32 2006/06/16 07:29:33 vruppert Exp $

Do you want to create a floppy disk image or a hard disk image?
Please type hd or fd. [hd] fd

Choose the size of floppy disk image to create, in megabytes.
Please type 0.16, 0.18, 0.32, 0.36, 0.72, 1.2, 1.44, 1.68, 1.72, or 2.88.
[1.44]
I will create a floppy image with
  cyl=80
  heads=2
  sectors per track=18
  total sectors=2880
  total bytes=1474560

What should I name the image?
[a.img] floppy.img

Writing: [] Done.

I wrote 1474560 bytes to floppy.img.

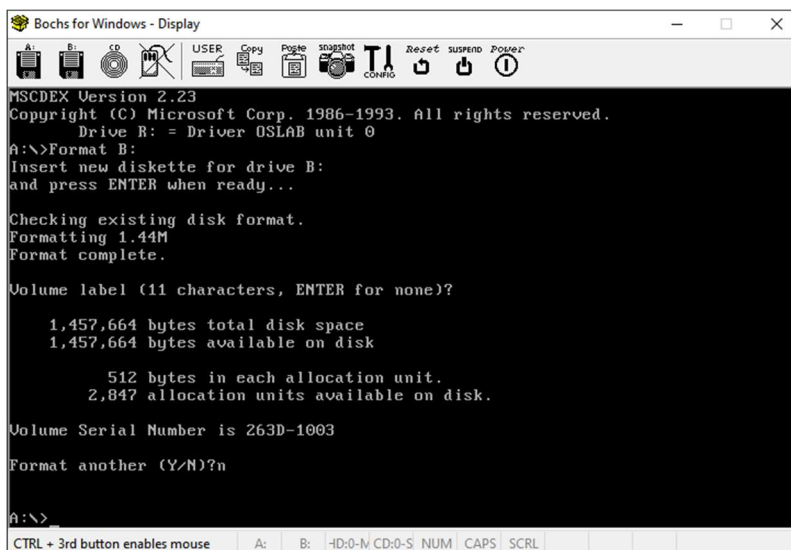
The following line should appear in your bochsrc:
  floppy: image="floppy.img", status=inserted
(The line is stored in your windows clipboard, use CTRL-V to paste)

Press any key to continue

C:\OS\LAB\LAB1>
```

Gambar 1.5 – 1.8 Membuat File image Floppy

### 4. Memformat floppya.img dengan DosFp



```
Bochs for Windows - Display

MSCDEX Version 2.23
Copyright (C) Microsoft Corp. 1986-1993. All rights reserved.
Drive R: = Driver OSLAB unit 0
A:\>Format B:
Insert new diskette for drive B:
and press ENTER when ready...

Checking existing disk format.
Formatting 1.44M
Format complete.

Volume label (11 characters, ENTER for none)?

1,457,664 bytes total disk space
1,457,664 bytes available on disk

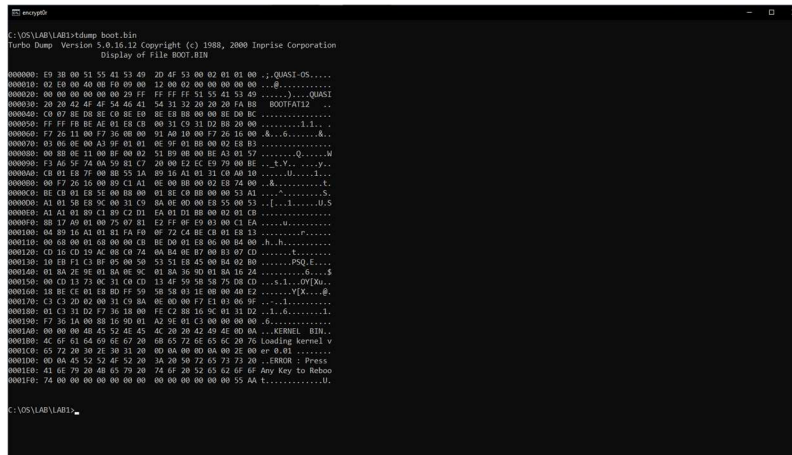
512 bytes in each allocation unit.
2,847 allocation units available on disk.

Volume Serial Number is 263D-1003
Format another (Y/N)?n

A:\>
```

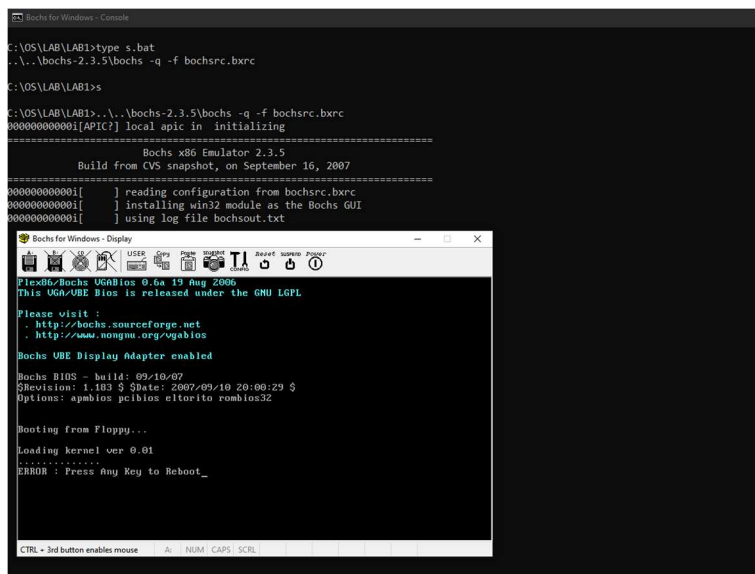
Gambar 1.11 Bochs PC-Simulator

5. Dump data dari image boot.bin dengan tdump



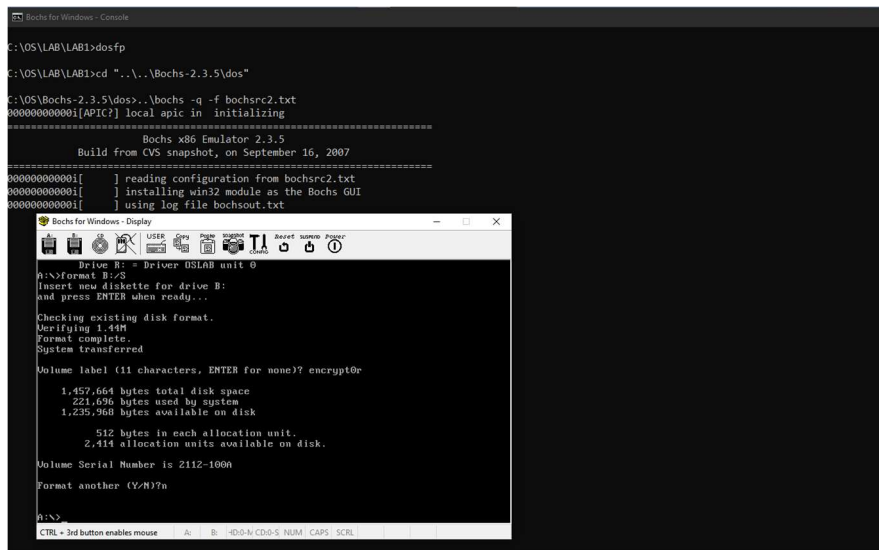
Gambar 1.14 Melihat data bootsector dengan program tdump

6. Booting dari floppy.img dengan s.bat



Gambar 1.15 isi File ‘s.bat’

## 7. Memformat floppy.img dan mengisi SystemFile dengan DosFp



```
C:\OS\LAB\LAB1>dosfp
C:\OS\LAB\LAB1>cd "..\..\Bochs-2.3.5\dos"
C:\Bochs-2.3.5\dos>.\bochs -q -f bochsrc2.txt
000000000000i[APIC?] local apic in  initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
000000000000i[ ] reading configuration from bochsrc2.txt
000000000000i[ ] installing win32 module as the Bochs GUI
000000000000i[ ] using log file bochsout.txt

Bochs for Windows - Display
Drive B: = Driver OS/2 unit 0
A:\>format B:/S
[Insert new diskette for Drive B:
and press ENTER when ready...]

Checking existing disk format.
Verifying 1.44M
Format complete.
System transferred

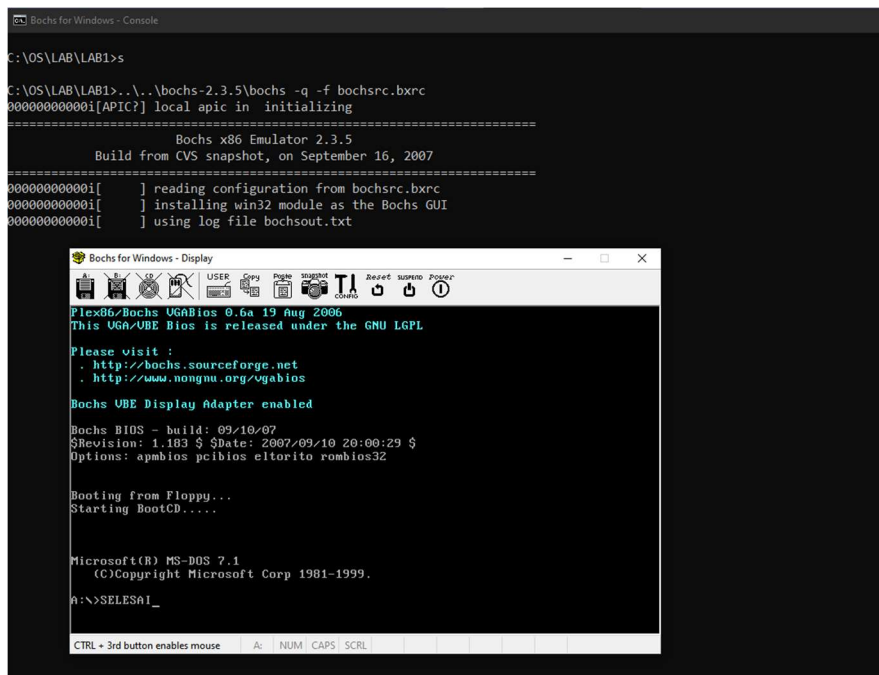
Volume label (11 characters, ENTER for none)? encrypt0r

1,457,664 bytes total disk space
221,696 bytes used by system
1,235,968 bytes available on disk

512 bytes in each allocation unit.
2,414 allocation units available on disk.

Volume Serial Number is 2112-1000
Format another (Y/N)?n
A:\>
```

## 8. Booting ulang dari floppy.img yang sudah terisi System File dengan s.bat



```
C:\OS\LAB\LAB1>s
C:\OS\LAB\LAB1>..\..\bochs-2.3.5\bochs -q -f bochsrc.bxrc
000000000000i[APIC?] local apic in  initializing
=====
Bochs x86 Emulator 2.3.5
Build from CVS snapshot, on September 16, 2007
=====
000000000000i[ ] reading configuration from bochsrc.bxrc
000000000000i[ ] installing win32 module as the Bochs GUI
000000000000i[ ] using log file bochsout.txt

Bochs for Windows - Display
Plex06/Bochs UGABios 0.6a 19 Aug 2006
This UGA/UBE Bios is released under the GNU LGPL

Please visit :
. http://bochs.sourceforge.net
. http://www.nongnu.org/vgabios

Bochs UBE Display Adapter enabled

Bochs BIOS - build: 09/10/07
$Revision: 1.183 $ $Date: 2007/09/10 20:00:29 $
Options: apmbios pcibios eltorito rombios32

Booting from Floppy...
Starting BootCD.....

Microsoft(R) MS-DOS 7.1
(C)Copyright Microsoft Corp 1981-1999.

A:\>SELESAI_
```

## 9. Selesai

## B. Tugas

1. ASCII (American Standard Code for Information Interchange) merupakan Kode Standar Amerika untuk Pertukaran Informasi atau sebuah standar internasional dalam pengkodean huruf dan simbol seperti Unicode dan Hex tetapi ASCII lebih bersifat universal. Dalam bahasa komputer 0 dan 1 tidak ada cara lain untuk mewakili huruf dan karakter yang bukan nomer. Semuanya harus menggunakan 0 dan 1. Salah satu jalan untuk berbahasa dengan komputer dengan cara menggunakan tabel ASCII. Tabel ASCII merupakan tabel atau daftar yang berisi semua huruf dalam alfabet romawi ditambah beberapa karakter tambahan. Dalam tabel ini setiap karakter akan selalu diwakili oleh sejumlah kode yang sama.

**Tabel Kode ASCII**

|    |    |     |    |    |       |    |    |   |     |    |   |     |    |     |
|----|----|-----|----|----|-------|----|----|---|-----|----|---|-----|----|-----|
| 0  | 00 | NUL | 26 | 1A | SUB   | 52 | 34 | 4 | 78  | 4E | N | 104 | 68 | h   |
| 1  | 01 | SOH | 27 | 1B | ESC   | 53 | 35 | 5 | 79  | 4F | O | 105 | 69 | i   |
| 2  | 02 | STX | 28 | 1C | FS    | 54 | 36 | 6 | 80  | 50 | P | 106 | 6A | j   |
| 3  | 03 | ETX | 29 | 1D | GS    | 55 | 37 | 7 | 81  | 51 | Q | 107 | 6B | k   |
| 4  | 04 | EOT | 30 | 1E | RS    | 56 | 38 | 8 | 82  | 52 | R | 108 | 6C | l   |
| 5  | 05 | ENQ | 31 | 1F | US    | 57 | 39 | 9 | 83  | 53 | S | 109 | 6D | m   |
| 6  | 06 | ACK | 32 | 20 | space | 58 | 3A | : | 84  | 54 | T | 110 | 6E | n   |
| 7  | 07 | BEL | 33 | 21 | !     | 59 | 3B | ; | 85  | 55 | U | 111 | 6F | o   |
| 8  | 08 | BS  | 34 | 22 | "     | 60 | 3C | < | 86  | 56 | V | 112 | 70 | p   |
| 9  | 09 | HT  | 35 | 23 | #     | 61 | 3D | = | 87  | 57 | W | 113 | 71 | q   |
| 10 | 0A | LF  | 36 | 24 | \$    | 62 | 3E | > | 88  | 58 | X | 114 | 72 | r   |
| 11 | 0B | VT  | 37 | 25 | %     | 63 | 3F | ? | 89  | 59 | Y | 115 | 73 | s   |
| 12 | 0C | FF  | 38 | 26 | &     | 64 | 40 | @ | 90  | 5A | Z | 116 | 74 | t   |
| 13 | 0D | CR  | 39 | 27 | '     | 65 | 41 | A | 91  | 5B | [ | 117 | 75 | u   |
| 14 | 0E | SO  | 40 | 28 | (     | 66 | 42 | B | 92  | 5C | \ | 118 | 76 | v   |
| 15 | 0F | SI  | 41 | 29 | )     | 67 | 43 | C | 93  | 5D | ] | 119 | 77 | w   |
| 16 | 10 | DLE | 42 | 2A | *     | 68 | 44 | D | 94  | 5E | ^ | 120 | 78 | x   |
| 17 | 11 | DC1 | 43 | 2B | +     | 69 | 45 | E | 95  | 5F | _ | 121 | 79 | y   |
| 18 | 12 | DC2 | 44 | 2C | ,     | 70 | 46 | F | 96  | 60 | ` | 122 | 7A | z   |
| 19 | 13 | DC3 | 45 | 2D | -     | 71 | 47 | G | 97  | 61 | a | 123 | 7B | {   |
| 20 | 14 | DC4 | 46 | 2E | .     | 72 | 48 | H | 98  | 62 | b | 124 | 7C |     |
| 21 | 15 | NAK | 47 | 2F | /     | 73 | 49 | I | 99  | 63 | c | 125 | 7D | }   |
| 22 | 16 | SYN | 48 | 30 | 0     | 74 | 4A | J | 100 | 64 | d | 126 | 7E | ~   |
| 23 | 17 | ETB | 49 | 31 | 1     | 75 | 4B | K | 101 | 65 | e | 127 | 7F | DEL |
| 24 | 18 | CAN | 50 | 32 | 2     | 76 | 4C | L | 102 | 66 | f |     |    |     |
| 25 | 19 | EM  | 51 | 33 | 3     | 77 | 4D | M | 103 | 67 | g |     |    |     |

## 2. INSTRUKSI ASSEMBLY PADA INTEL KELUARGA x86



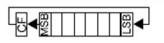
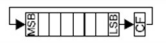


Intel Assembler 80186 and higher

### CodeTable 1/2



© 1996-2003 by Roger Jegerlehner, Switzerland  
V 2.3 English. Also available in Spanish

| TRANSFER     |                            |                  |   | Flags |   |   |   |   |   |   |   |   |
|--------------|----------------------------|------------------|---|-------|---|---|---|---|---|---|---|---|
| Name         | Comment                    | Code             | Operation                                       | O     | D | I | T | S | Z | A | P | C |
| MOV          | Move (copy)                | MOV Dest,Source  | Dest:=Source                                    |       |   |   |   |   |   |   |   |   |
| XCHG         | Exchange                   | XCHG Op1,Op2     | Op1:=Op2 , Op2:=Op1                             |       |   |   |   |   |   |   |   |   |
| STC          | Set Carry                  | STC              | CF:=1   |       |   |   |   |   |   |   |   | 1 |
| CLC          | Clear Carry                | CLC              | CF:=0   |       |   |   |   |   |   |   |   | 0 |
| CMC          | Complement Carry           | CMC              | CF:= ¬CF  |       |   |   |   |   |   |   |   | ± |
| STD          | Set Direction              | STD              | DF:=1 (string op's downwards)                   |       | 1 |   |   |   |   |   |   |   |
| CLD          | Clear Direction            | CLD              | DF:=0 (string op's upwards)                     |       | 0 |   |   |   |   |   |   |   |
| STI          | Set Interrupt              | STI              | IF:=1   |       |   | 1 |   |   |   |   |   |   |
| CLI          | Clear Interrupt            | CLI              | IF:=0   |       |   | 0 |   |   |   |   |   |   |
| PUSH         | Push onto stack            | PUSH Source      | DEC SP, [SP]:=Source                            |       |   |   |   |   |   |   |   |   |
| PUSHF        | Push flags                 | PUSHF            | O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL   |       |   |   |   |   |   |   |   |   |
| PUSHA        | Push all general registers | PUSHA            | AX, CX, DX, BX, SP, BP, SI, DI                  |       |   |   |   |   |   |   |   |   |
| POP          | Pop from stack             | POP Dest         | Dest:=[SP], INC SP                              |       |   |   |   |   |   |   |   |   |
| POPF         | Pop flags                  | POPF             | O, D, I, T, S, Z, A, P, C 286+: also NT, IOPL   | ±     | ± | ± | ± | ± | ± | ± | ± | ± |
| POPA         | Pop all general registers  | POPA             | DI, SI, BP, SP, BX, DX, CX, AX                  |       |   |   |   |   |   |   |   |   |
| CBW          | Convert byte to word       | CBW              | AX:=AL (signed)                                 |       |   |   |   |   |   |   |   |   |
| CWD          | Convert word to double     | CWD              | DX:AX:=AX (signed)                              | ±     |   |   |   |   | ± | ± | ± | ± |
| CWDE         | Conv word extended double  | CWDE 386         | EAX:=AX (signed)                                |       |   |   |   |   |   |   |   |   |
| IN <i>i</i>  | Input                      | IN Dest, Port    | AL/AX/EAX := byte/word/double of specified port |       |   |   |   |   |   |   |   |   |
| OUT <i>i</i> | Output                     | OUT Port, Source | Byte/word/double of specified port := AL/AX/EAX |       |   |   |   |   |   |   |   |   |

*i* for more information see instruction specifications      Flags: ±=affected by this instruction ?=undefined after this instruction

| ARITHMETIC    |                               |                 |  | Flags    |   |   |   |   |   |   |   |   |
|---------------|-------------------------------|-----------------|--|----------|---|---|---|---|---|---|---|---|
| Name          | Comment                       | Code            | Operation  | O        | D | I | T | S | Z | A | P | C |
| ADD           | Add                           | ADD Dest,Source | Dest:=Dest+Source  | ±        |   |   |   |   | ± | ± | ± | ± |
| ADC           | Add with Carry                | ADC Dest,Source | Dest:=Dest+Source+CF   | ±        |   |   |   |   | ± | ± | ± | ± |
| SUB           | Subtract                      | SUB Dest,Source | Dest:=Dest-Source  | ±        |   |   |   |   | ± | ± | ± | ± |
| SBB           | Subtract with borrow          | SBB Dest,Source | Dest:=Dest-(Source+CF)   | ±        |   |   |   |   | ± | ± | ± | ± |
| DIV           | Divide (unsigned)             | DIV Op          | Op=byte: AL:=AX / Op      AH:=Rest   | ?        |   |   |   |   | ? | ? | ? | ? |
| DIV           | Divide (unsigned)             | DIV Op          | Op=word: AX:=DX:AX / Op      DX:=Rest  | ?        |   |   |   |   | ? | ? | ? | ? |
| DIV 386       | Divide (unsigned)             | DIV Op          | Op=doublew.: EAX:=EDX:EAX / Op      EDX:=Rest  | ?        |   |   |   |   | ? | ? | ? | ? |
| IDIV          | Signed Integer Divide         | IDIV Op         | Op=byte: AL:=AX / Op      AH:=Rest   | ?        |   |   |   |   | ? | ? | ? | ? |
| IDIV          | Signed Integer Divide         | IDIV Op         | Op=word: AX:=DX:AX / Op      DX:=Rest  | ?        |   |   |   |   | ? | ? | ? | ? |
| IDIV 386      | Signed Integer Divide         | IDIV Op         | Op=doublew.: EAX:=EDX:EAX / Op      EDX:=Rest  | ?        |   |   |   |   | ? | ? | ? | ? |
| MUL           | Multiply (unsigned)           | MUL Op          | Op=byte: AX:=AL*Op      if AH=0 ♦  | ±        |   |   |   |   | ? | ? | ? | ± |
| MUL           | Multiply (unsigned)           | MUL Op          | Op=word: DX:AX:=AX*Op      if DX=0 ♦   | ±        |   |   |   |   | ? | ? | ? | ± |
| MUL 386       | Multiply (unsigned)           | MUL Op          | Op=double: EDX:EAX:=EAX*Op      if EDX=0 ♦   | ±        |   |   |   |   | ? | ? | ? | ± |
| IMUL <i>i</i> | Signed Integer Multiply       | IMUL Op         | Op=byte: AX:=AL*Op      if AL sufficient ♦   | ±        |   |   |   |   | ? | ? | ? | ± |
| IMUL          | Signed Integer Multiply       | IMUL Op         | Op=word: DX:AX:=AX*Op      if AX sufficient ♦  | ±        |   |   |   |   | ? | ? | ? | ± |
| IMUL 386      | Signed Integer Multiply       | IMUL Op         | Op=double: EDX:EAX:=EAX*Op      if EAX sufficient ♦                                  | ±        |   |   |   |   | ? | ? | ? | ± |
| INC           | Increment                     | INC Op          | Op:=Op+1 (Carry not affected !)  | ±        |   |   |   |   | ± | ± | ± | ± |
| DEC           | Decrement                     | DEC Op          | Op:=Op-1 (Carry not affected !)  | ±        |   |   |   |   | ± | ± | ± | ± |
| CMP           | Compare                       | CMP Op1,Op2     | Op1-Op2  | ±        |   |   |   |   | ± | ± | ± | ± |
| SAL           | Shift arithmetic left (≡ SHL) | SAL Op,Quantity |   | <i>i</i> |   |   |   |   | ± | ± | ? | ± |
| SAR           | Shift arithmetic right        | SAR Op,Quantity |  | <i>i</i> |   |   |   |   | ± | ± | ? | ± |
| RCL           | Rotate left through Carry     | RCL Op,Quantity |   | <i>i</i> |   |   |   |   |   |   |   | ± |
| RCR           | Rotate right through Carry    | RCR Op,Quantity |  | <i>i</i> |   |   |   |   |   |   |   | ± |
| ROL           | Rotate left                   | ROL Op,Quantity |   | <i>i</i> |   |   |   |   |   |   |   | ± |
| ROR           | Rotate right                  | ROR Op,Quantity |  | <i>i</i> |   |   |   |   |   |   |   | ± |

*i* for more information see instruction specifications      ♦ then CF:=0, OF:=0 else CF:=1, OF:=1

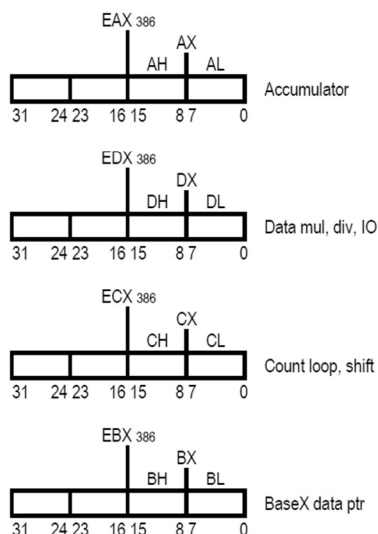
| LOGIC |                               |                 |   | Flags  |   |   |   |   |   |   |   |   |
|-------|-------------------------------|-----------------|---|--|---|---|---|---|---|---|---|---|
| Name  | Comment                       | Code            | Operation   | O  | D | I | T | S | Z | A | P | C |
| NEG   | Negate (two-complement)       | NEG Op          | Op:=0-Op      if Op=0 then CF:=0 else CF:=1   | ±  |   |   |   |   | ± | ± | ± | ± |
| NOT   | Invert each bit               | NOT Op          | Op:=¬Op (invert each bit)   |  |   |   |   |   |   |   |   |   |
| AND   | Logical and                   | AND Dest,Source | Dest:=Dest∧Source   | 0  |   |   |   |   | ± | ± | ? | 0 |
| OR    | Logical or                    | OR Dest,Source  | Dest:=Dest∨Source   | 0  |   |   |   |   | ± | ± | ? | 0 |
| XOR   | Logical exclusive or          | XOR Dest,Source | Dest:=Dest (exor) Source  | 0  |   |   |   |   | ± | ± | ? | 0 |
| SHL   | Shift logical left    (≡ SAL) | SHL Op,Quantity |  | i  |   |   |   |   | ± | ± | ? | ± |
| SHR   | Shift logical right           | SHR Op,Quantity |   |  | i |   |   |   |   | ± | ± | ? |

| MISC |                        |                 |  | Flags |   |   |   |   |   |   |   |   |
|------|------------------------|-----------------|--|-------|---|---|---|---|---|---|---|---|
| Name | Comment                | Code            | Operation  | O     | D | I | T | S | Z | A | P | C |
| NOP  | No operation           | NOP             | No operation                                       |       |   |   |   |   |   |   |   |   |
| LEA  | Load effective address | LEA Dest,Source | Dest := address of Source                          |       |   |   |   |   |   |   |   |   |
| INT  | Interrupt              | INT Nr          | interrupts current program, runs spec. int-program |       |   | 0 | 0 |   |   |   |   |   |

| JUMPS (flags remain unchanged) |                              |           |           | Name  | Comment                        | Code       | Operation      |
|--------------------------------|------------------------------|-----------|-----------|-------|--------------------------------|------------|----------------|
| Name                           | Comment                      | Code      | Operation | Name  | Comment                        | Code       | Operation      |
| CALL                           | Call subroutine              | CALL Proc |           | RET   | Return from subroutine         | RET        |                |
| JMP                            | Jump                         | JMP Dest  |           |       |                                |            |                |
| JE                             | Jump if Equal                | JE Dest   | (= JZ)    | JNE   | Jump if not Equal              | JNE Dest   | (= JNZ)        |
| JZ                             | Jump if Zero                 | JZ Dest   | (= JE)    | JNZ   | Jump if not Zero               | JNZ Dest   | (= JNE)        |
| JCXZ                           | Jump if CX Zero              | JCXZ Dest |           | JECXZ | Jump if ECX Zero               | JECXZ Dest | <sup>386</sup> |
| JP                             | Jump if Parity (Parity Even) | JP Dest   | (= JPE)   | JNP   | Jump if no Parity (Parity Odd) | JNP Dest   | (= JPO)        |
| JPE                            | Jump if Parity Even          | JPE Dest  | (= JP)    | JPO   | Jump if Parity Odd             | JPO Dest   | (= JNP)        |

| JUMPS Unsigned (Cardinal) |                            |           |               | JUMPS Signed (Integer) |                              |           |           |
|---------------------------|----------------------------|-----------|---------------|------------------------|------------------------------|-----------|-----------|
| Name                      | Comment                    | Code      | Operation     | Name                   | Comment                      | Code      | Operation |
| JA                        | Jump if Above              | JA Dest   | (= JNBE)      | JG                     | Jump if Greater              | JG Dest   | (= JNLE)  |
| JAE                       | Jump if Above or Equal     | JAE Dest  | (= JNB = JNC) | JGE                    | Jump if Greater or Equal     | JGE Dest  | (= JNL)   |
| JB                        | Jump if Below              | JB Dest   | (= JNAE = JC) | JL                     | Jump if Less                 | JL Dest   | (= JNGE)  |
| JBE                       | Jump if Below or Equal     | JBE Dest  | (= JNA)       | JLE                    | Jump if Less or Equal        | JLE Dest  | (= JNG)   |
| JNA                       | Jump if not Above          | JNA Dest  | (= JBE)       | JNG                    | Jump if not Greater          | JNG Dest  | (= JLE)   |
| JNAE                      | Jump if not Above or Equal | JNAE Dest | (= JB = JC)   | JNGE                   | Jump if not Greater or Equal | JNGE Dest | (= JL)    |
| JNB                       | Jump if not Below          | JNB Dest  | (= JAE = JNC) | JNL                    | Jump if not Less             | JNL Dest  | (= JGE)   |
| JNBE                      | Jump if not Below or Equal | JNBE Dest | (= JA)        | JNLE                   | Jump if not Less or Equal    | JNLE Dest | (= JG)    |
| JC                        | Jump if Carry              | JC Dest   |               | JO                     | Jump if Overflow             | JO Dest   |           |
| JNC                       | Jump if no Carry           | JNC Dest  |               | JNO                    | Jump if no Overflow          | JNO Dest  |           |
|                           |                            |           |               | JS                     | Jump if Sign (= negative)    | JS Dest   |           |
|                           |                            |           |               | JNS                    | Jump if no Sign (= positive) | JNS Dest  |           |

## General Registers:

Flags: 

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | - | - | - | O | D | I | T | S | Z | - | A | - | P | - | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

## Control Flags (how instructions are carried out):

D: Direction 1 = string op's process down from high to low address  
 I: Interrupt whether interrupts can occur. 1 = enabled  
 T: Trap single step for debugging

## Example:

```

.DOSSEG ; Demo program
.MODEL SMALL
.STACK 1024

Two EQU 2 ; Const
.DATA
VarB DB ? ; define Byte, any value
VarW DW 1010b ; define Word, binary
VarW2 DW 257 ; define Word, decimal
VarD DD 0AFFFFh ; define Doubleword, hex
S DB "Hello!",0 ; define String
.CODE
main: MOV AX,DGROUP ; resolved by linker
      MOV DS,AX ; init datasegment reg
      MOV [VarB],42 ; init VarB
      MOV [VarD],-7 ; set VarD
      MOV BX,Offset[S] ; addr of "H" of "Hello!"
      MOV AX,[VarW] ; get value into accumulator
      ADD AX,[VarW2] ; add VarW2 to AX
      MOV [VarW2],AX ; store AX in VarW2
      MOV AX,4C00h ; back to system
      INT 21h
      END main

```



## Status Flags (result of operations):

C: Carry result of unsigned op. is too large or below zero. 1 = carry/borrow  
 O: Overflow result of signed op. is too large or small. 1 = overflow/underflow  
 S: Sign sign of result. Reasonable for Integer only. 1 = neg. / 0 = pos.  
 Z: Zero result of operation is zero. 1 = zero  
 A: Aux. carry similar to Carry but restricted to the low nibble only  
 P: Parity 1 = result has even number of set bits