Nama : Muhammad Rafi

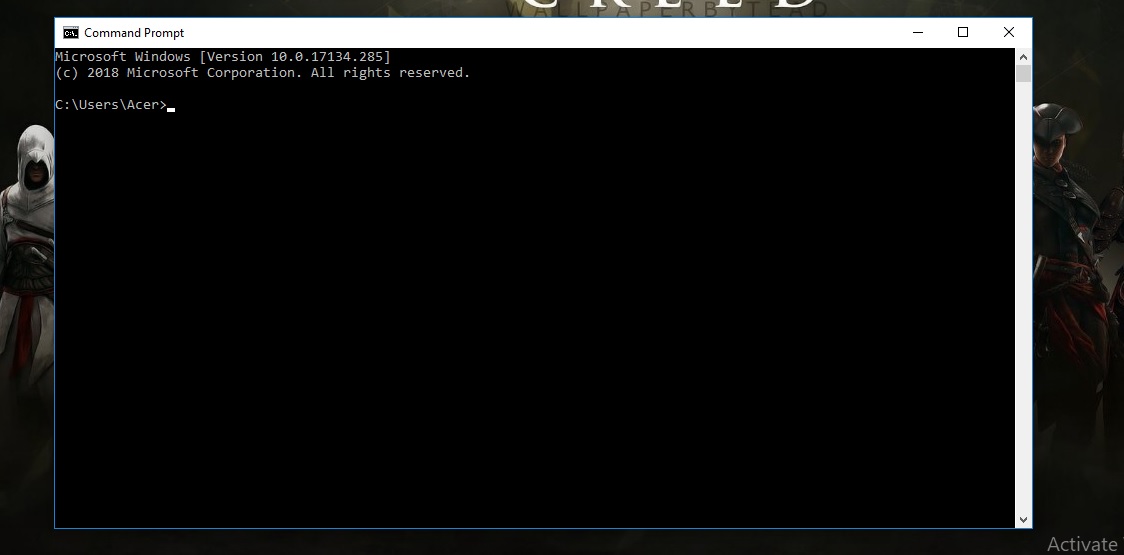
NIM : L200174138

Kelas : Internasional (X) 2017

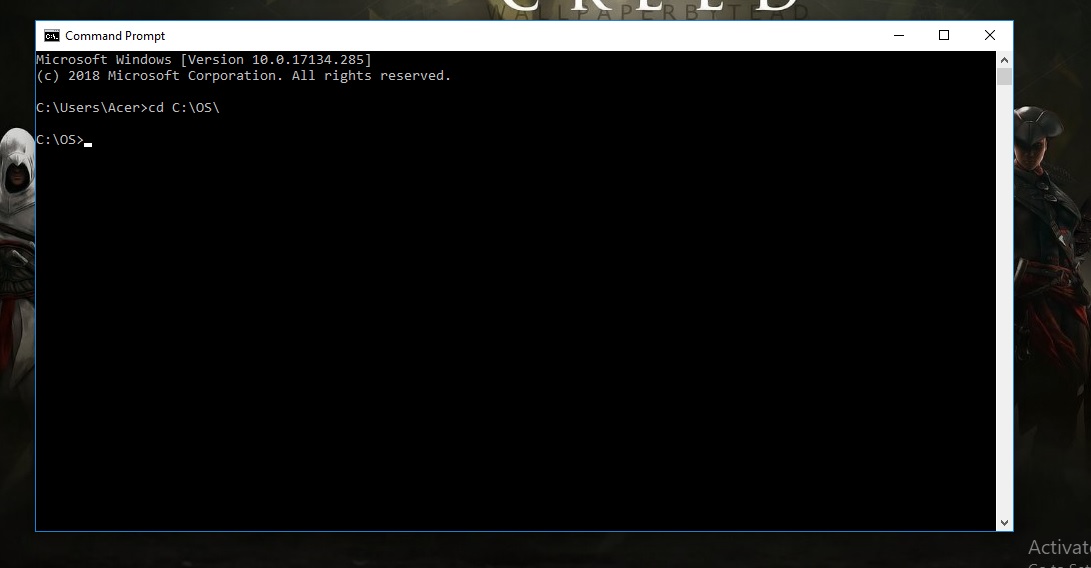
Praktikum 1

**1.Menuju ke direktori kerja**

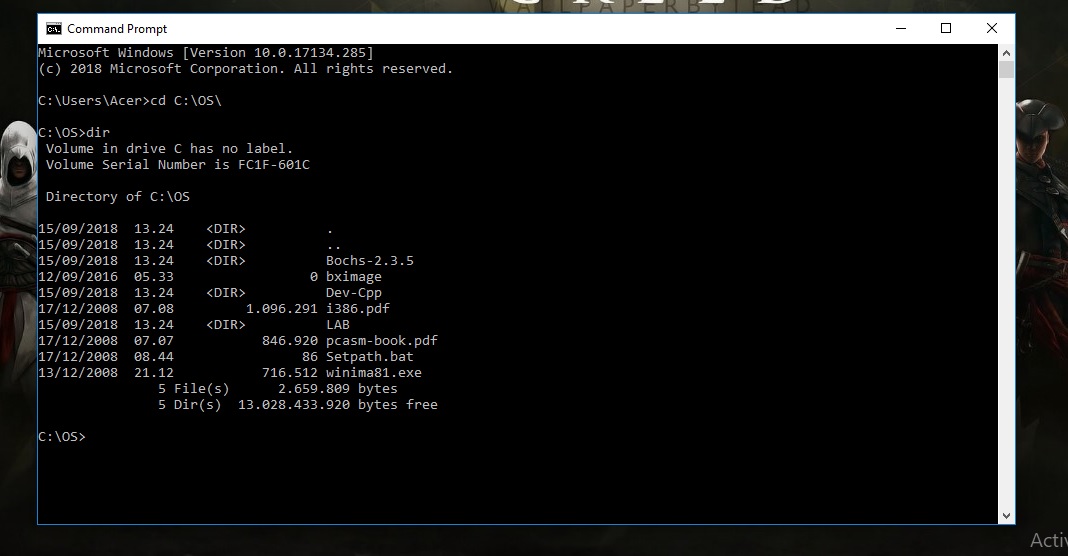
A.MENJALANKAN “**CMD”** (COMMAND PROMPT)



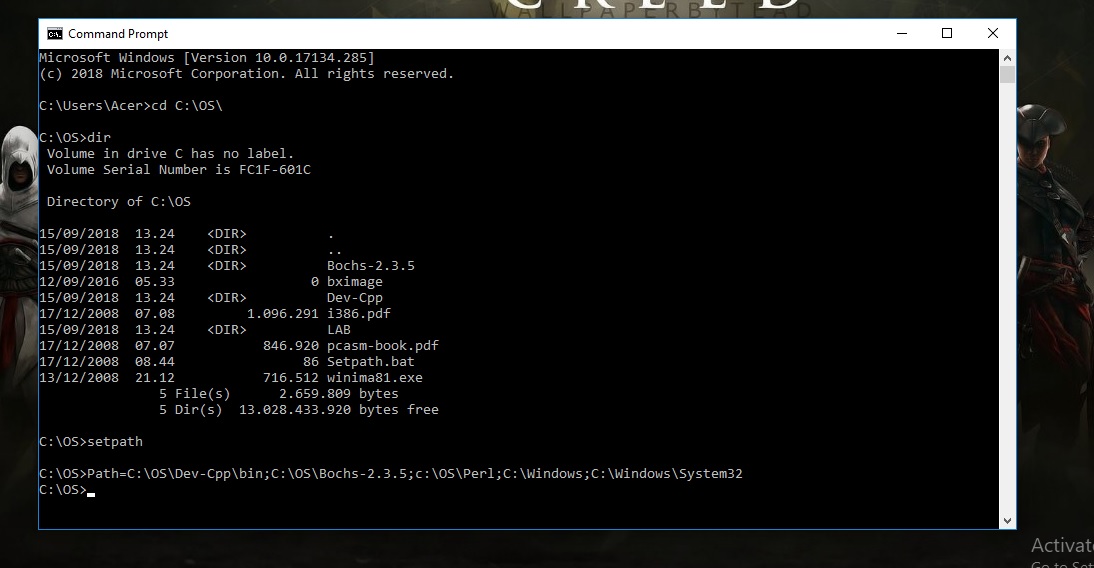
B.MASUK KE DIREKTORI KERJA **“C:\OS”**



C.MEMASUKKAN PERINTAH **“DIR”** (DIRECTORY) UNTUK MELIHAT ISI DIREKTORI DI DALAM FILE TERSEBUT.

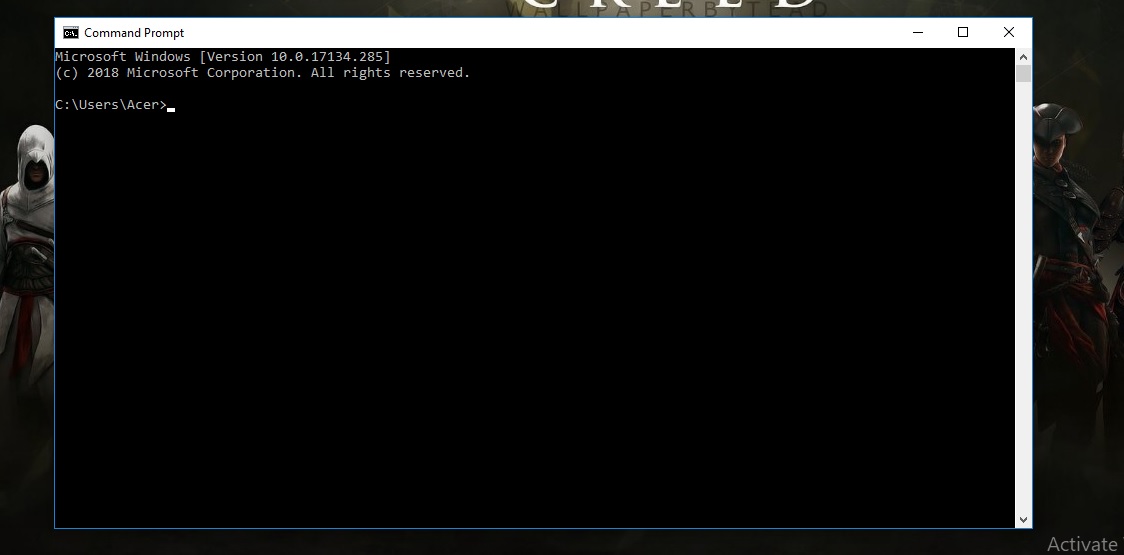


D.MENJALANKAN **“SETPATH”**

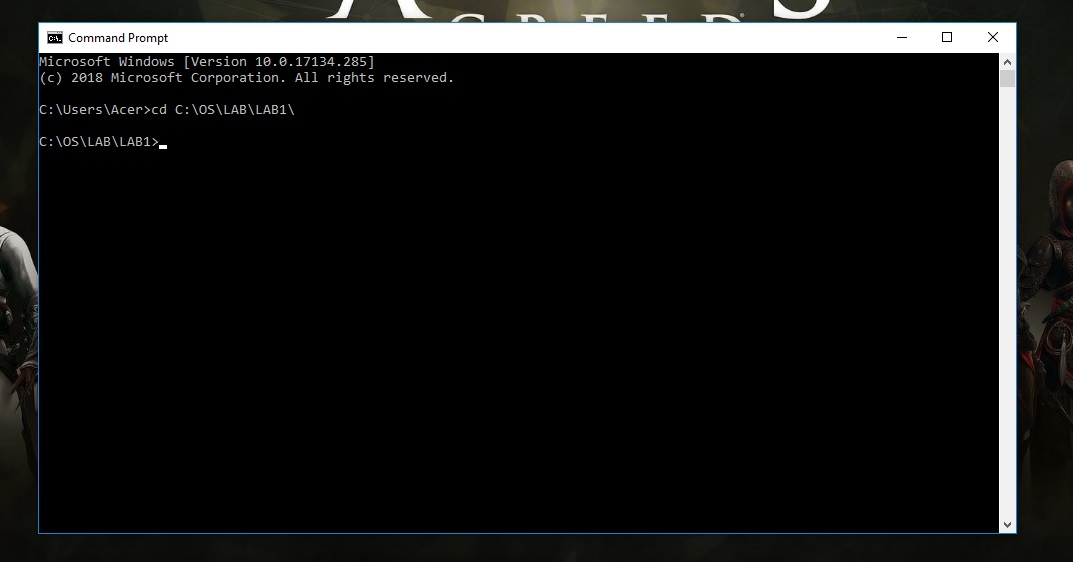
**

**2.Melihat isi direktori kerja**

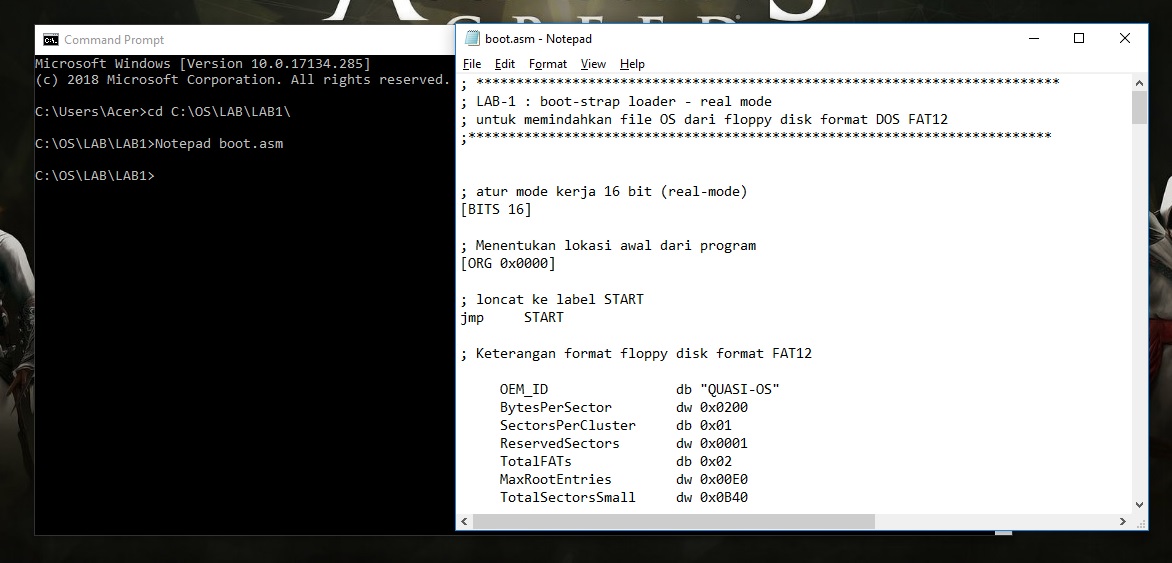
A.JALANKAN **“CMD”**



B.MASUK KE DIREKTORI KERJA **“C:\OS\LAB\LAB1”**

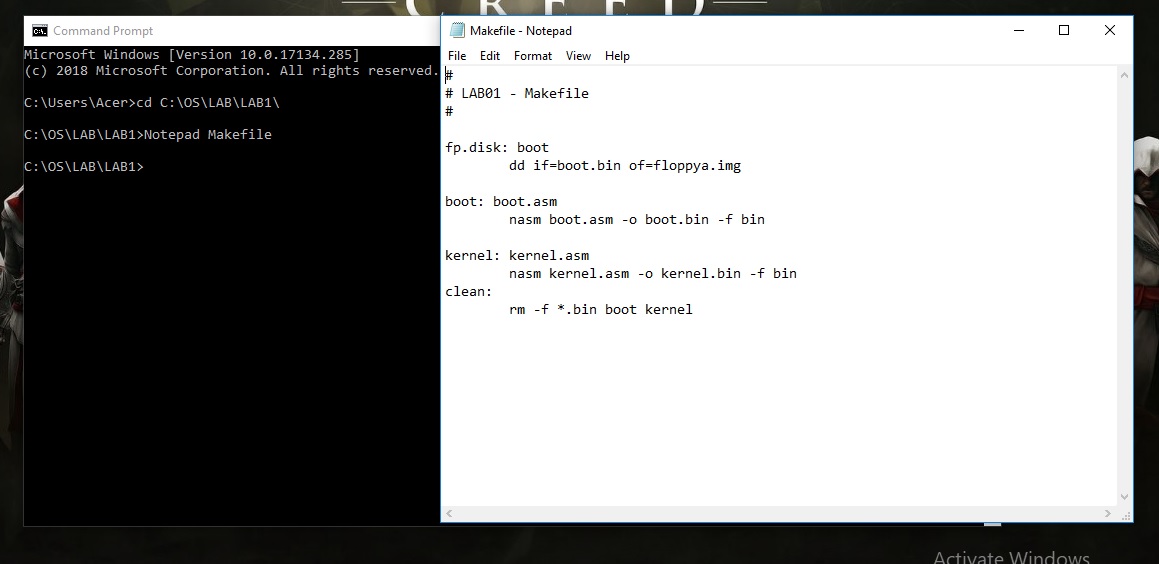


C.MEMASUKKAN PERINTAH **“NOTEPAD BOOT.ASM”** UNTUK MEMBUKA FILE

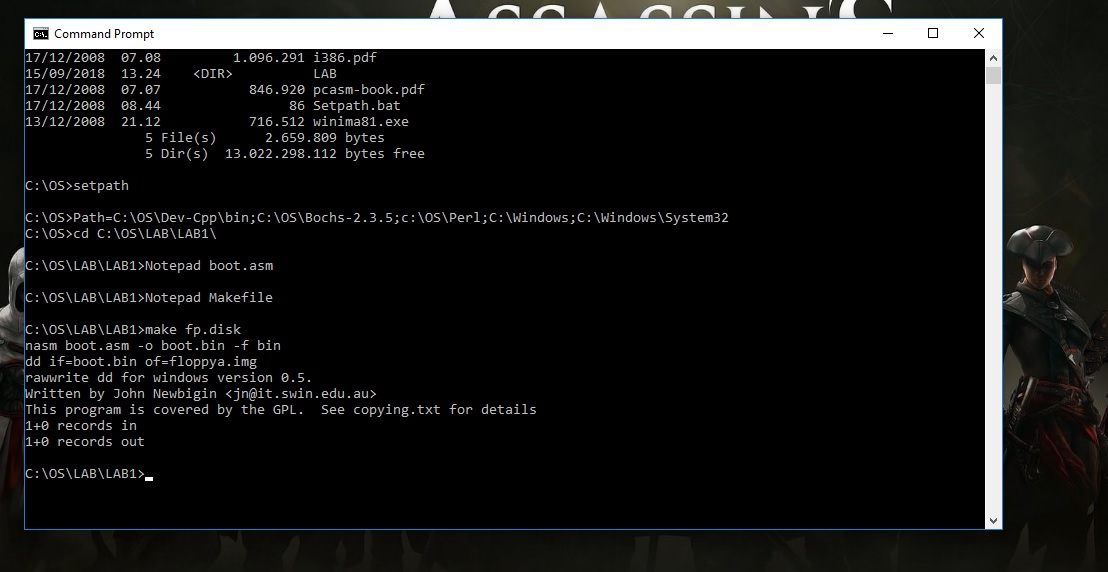


**3.Makefile**

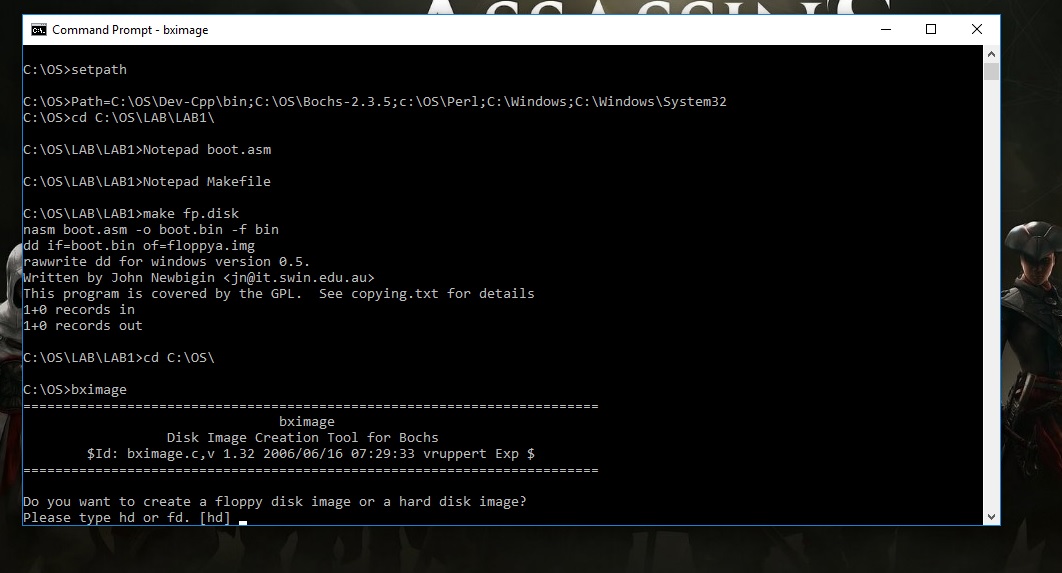
A.MEMBUKA **“NOTEPAD MAKEFILE”** PADA DIREKTORI KERJA **“C:\OS\LAB\LAB1”**



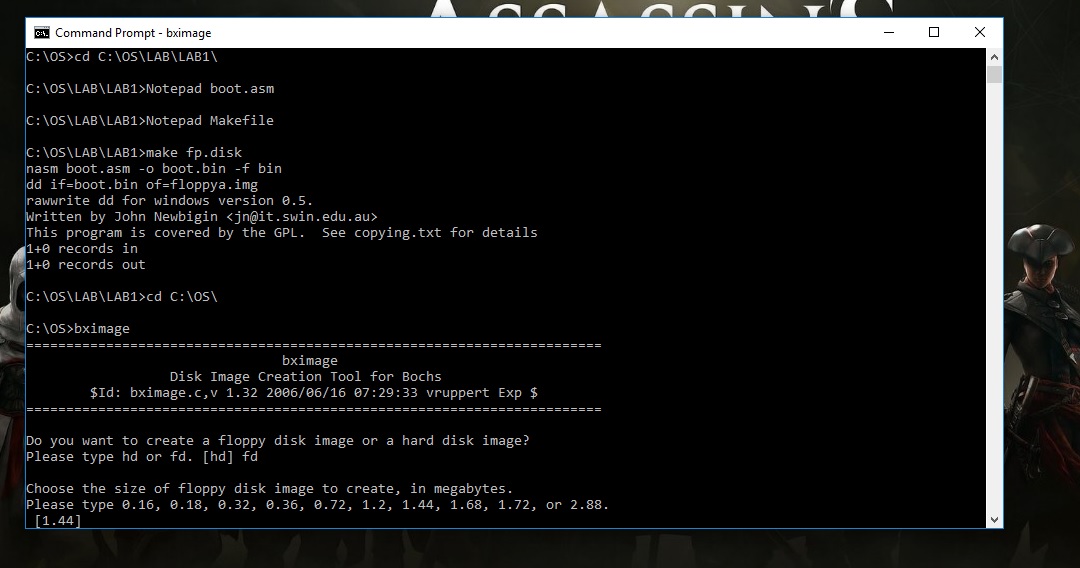
B.MENJALANKAN PERINTAH **“MAKE FP.DISK”**



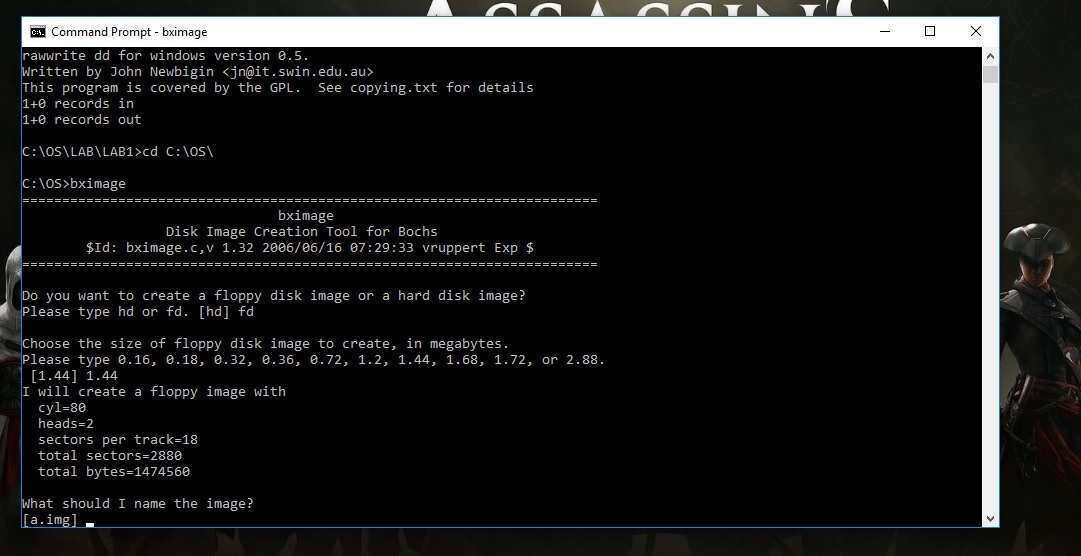
**4.Boot Disk**

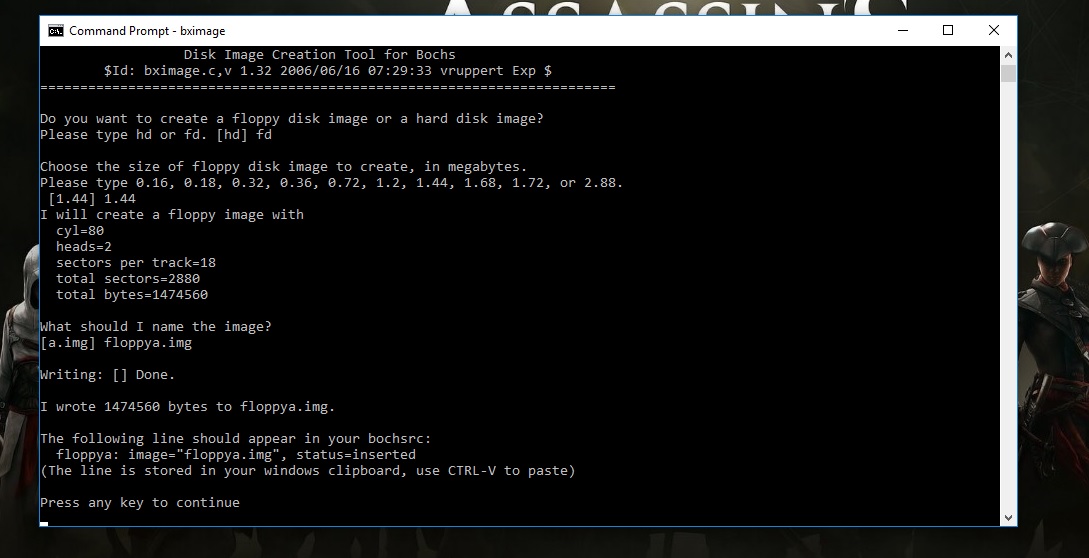
A.MEMANGGIL PERINTAH **“BXIMAGE”**

B.MEMILIH DISK YG AKAN DIGUNAKAN UNTUK MEMBUAT FLOPPY IMAGE



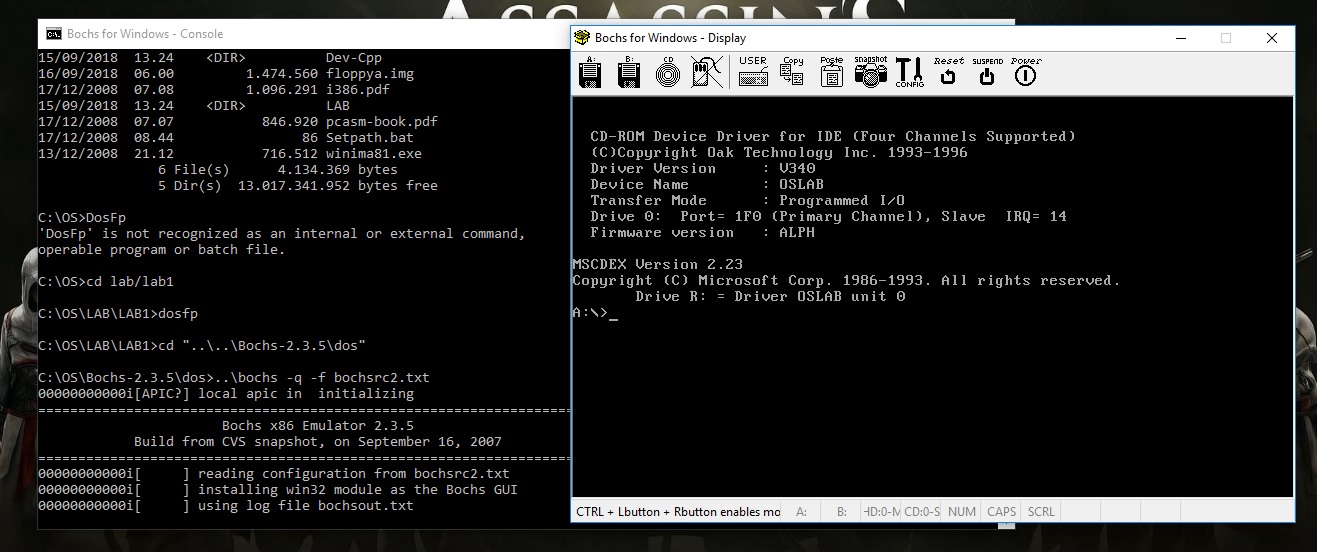
C.MENENTUKAN KAPASITAS YG AKAN DIGUNAKAN



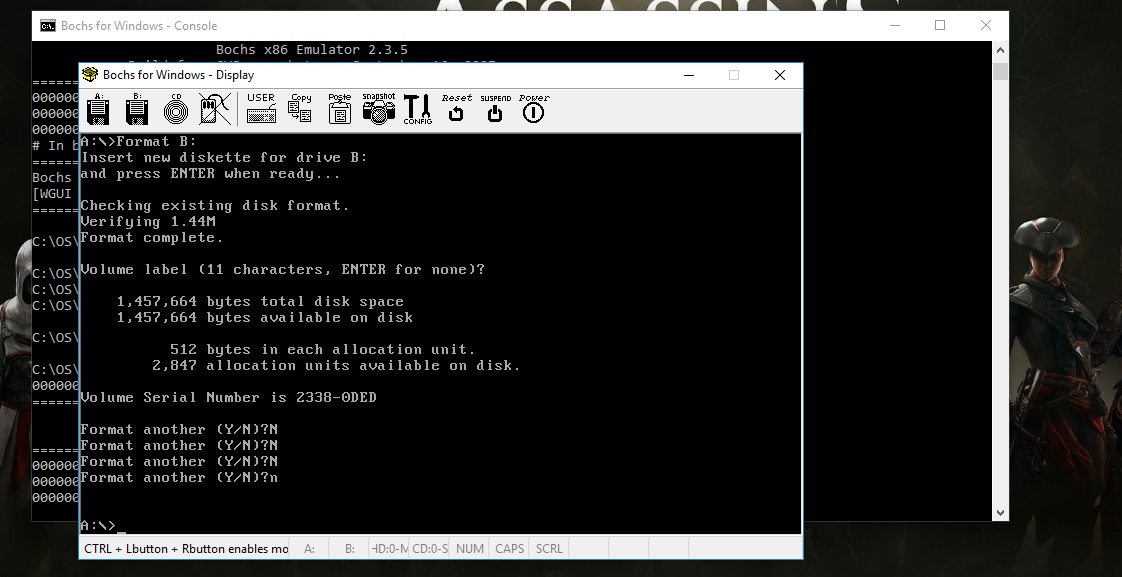
D.MEMBERI NAMA PADA FILE

**5.Formatted Disk**

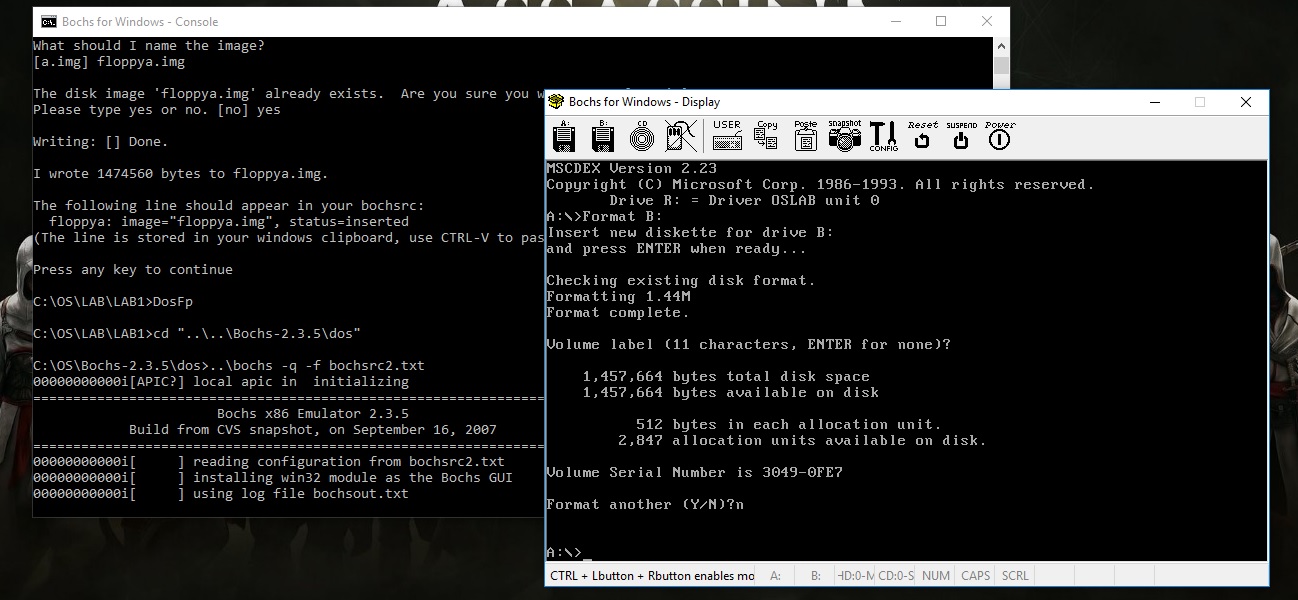
A.MENJALANKAN **“DOSFP”**



B.FORMAT **“DRIVE B:”**

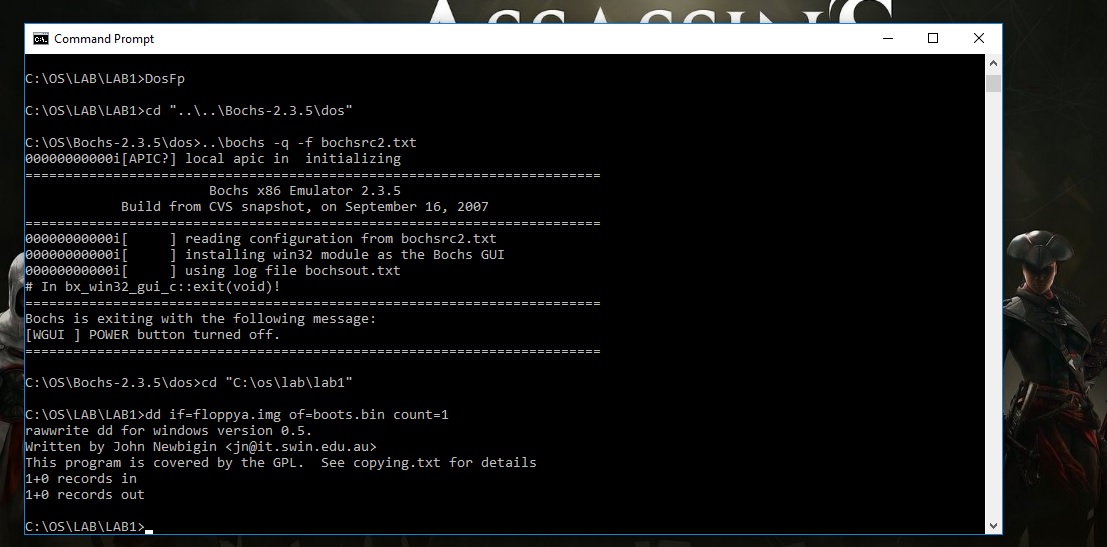


C.MENUTUP PC-SIMULATOR

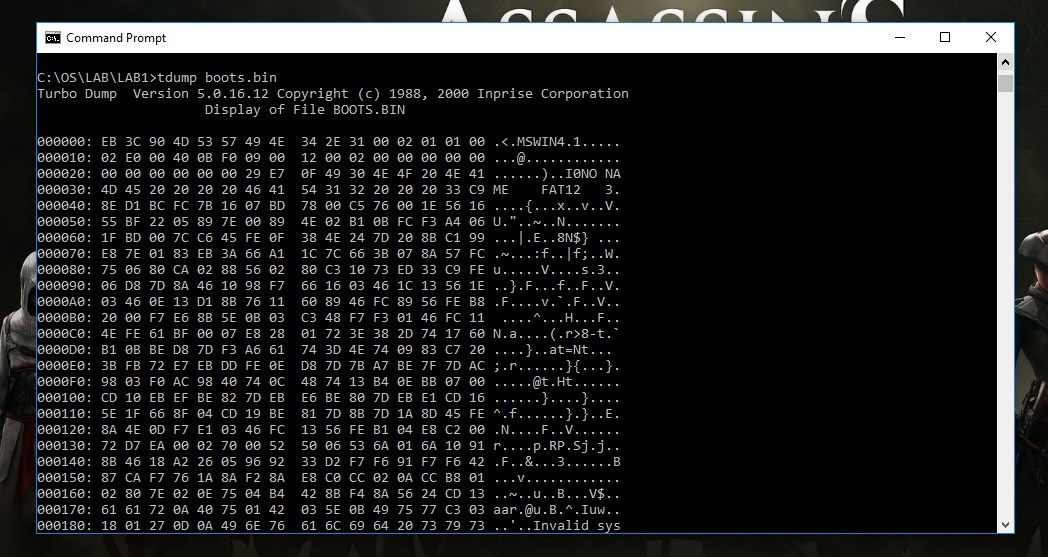


**6.Melihat data dalam boot sector**

A.MEMASUKKAN PERINTAH **“DD IF=FLOPPYA.IMG OF=BOOTS.BIN COUNT=1”**

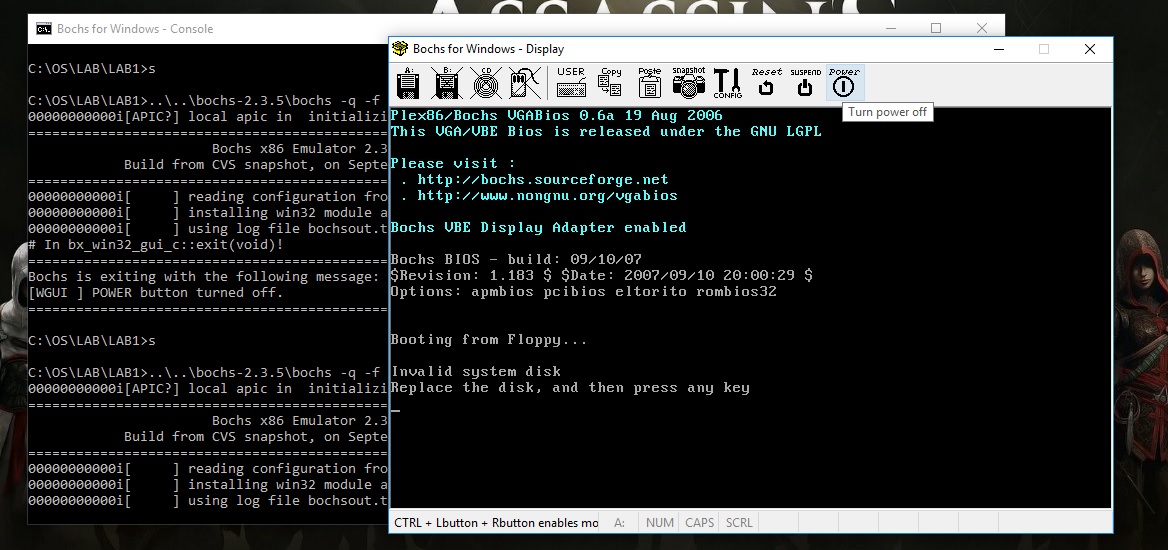


B.MENJALANKAN PERINTAH **“TDUMP BOOTS.BIN”**

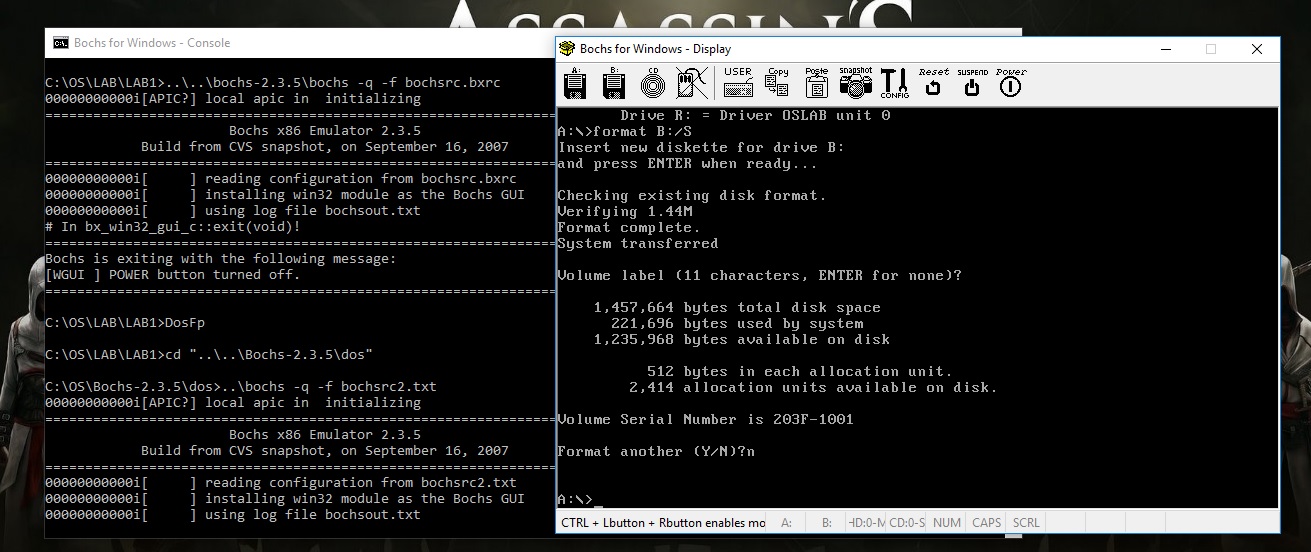


**7.”Boot” PC-Simulator dengan file image “floppya.img”**

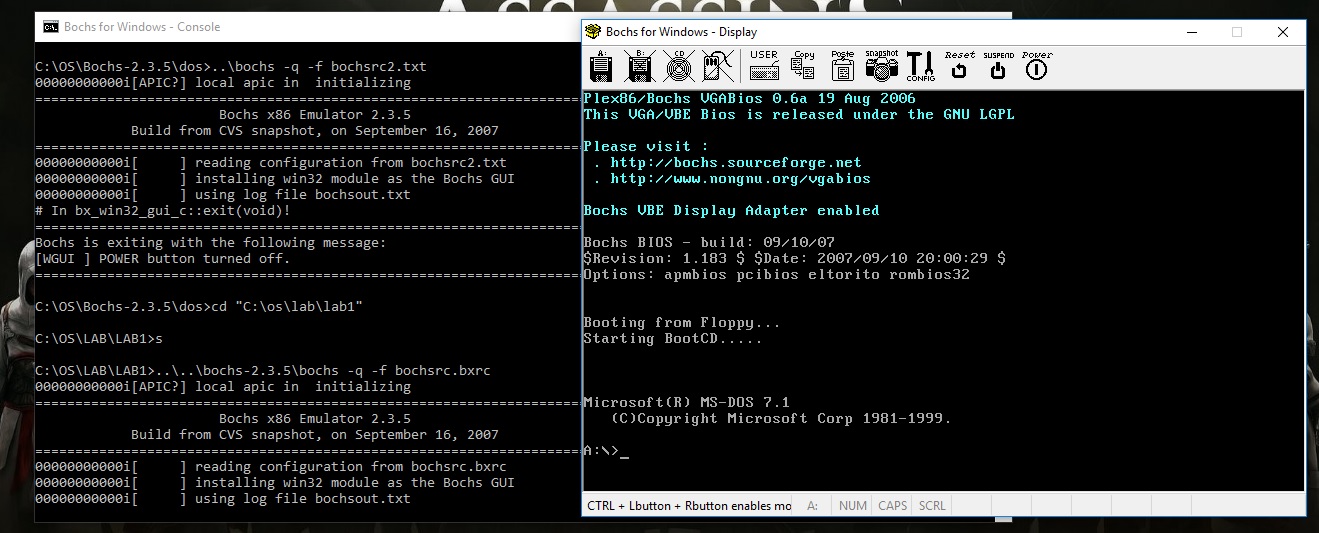
A.MELIHAT ISI FILE **“S.BAT”**



B.MEMASUKKAN PERINTAH **“A:>FORMAT B:/S”** PADA WINDWOS **“BOCHS”**

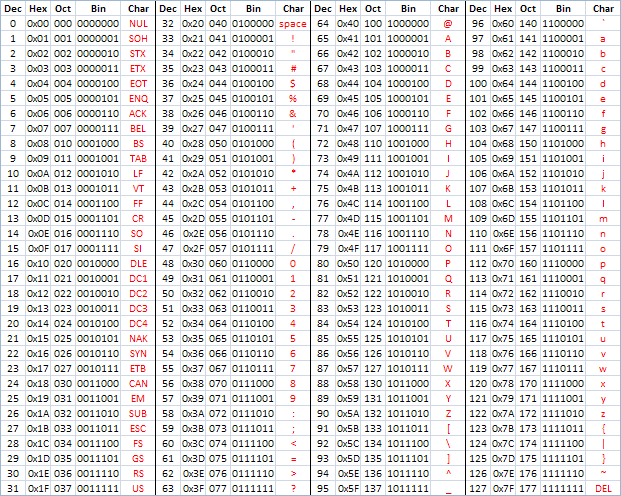


C.JIKA TIDAK ADA KESALAHAN MAKA **“BOCHS”** AKAN MENAMPILKAN :



**1. American Standard Code for Information Interchange (ASCII)** merupakansuatu standar internasional dalam kode [huruf](https://id.wikipedia.org/wiki/Huruf) dan [simbol](https://id.wikipedia.org/wiki/Simbol) seperti [Hex](https://id.wikipedia.org/wiki/Hex) dan [Unicode](https://id.wikipedia.org/wiki/Unicode) tetapi ASCII lebih bersifat universal, Ia selalu digunakan oleh [komputer](https://id.wikipedia.org/wiki/Komputer) dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan [biner](https://id.wikipedia.org/wiki/Biner) sebanyak 7 bit. Namun, ASCII disimpan sebagai sandi 8 bit dengan menambakan satu angka 0 sebagai bit significant paling tinggi. Bit tambahan ini sering digunakan untuk uji prioritas. Karakter control pada ASCII dibedakan menjadi 5 kelompok sesuai dengan penggunaan yaitu berturut-turut meliputi logical communication, Device control, Information separator, Code extention, dan physical communication. Code ASCII ini banyak dijumpai pada papan ketik (keyboard) computer atau instrument-instrument digital.

**Tabel ASCII:**



**Bahasa Assembley (x86) :**

**Daftar Assembly Directive**

|  |  |
| --- | --- |
| **Assembly Directive** | **Keterangan** |
| EQU | Pendefinisian konstanta |
| DB | Pendefinisian data dengan ukuran satuan 1 byte |
| DW | Pendefinisian data dengan ukuran satuan 1 word |
| DBIT | Pendefinisian data dengan ukuran satuan 1 bit |
| DS | Pemesanan tempat penyimpanan data di RAM |
| ORG | Inisialisasi alamat mulai program |
| END | Penanda akhir program |
| CSEG | Penanda penempatan di code segment |
| XSEG | Penanda penempatan di external data segment |
| DSEG | Penanda penempatan di internal direct data segment |
| ISEG | Penanda penempatan di internal indirect data segment |
| BSEG | Penanda penempatan di bit data segment |
| CODE | Penanda mulai pendefinisian program |
| XDATA | Pendefinisian external data |
| DATA | Pendefinisian internal direct data |
| IDATA | Pendefinisian internal indirect data |
| BIT | Pendefinisian data bit |
| #INCLUDE | Mengikutsertakan file program lain |

**Daftar Instruksi**

|  |  |
| --- | --- |
| **Instruksi** | **Keterangan Singkatan** |
| ACALL | Absolute Call |
| ADD | Add |
| ADDC | Add with Carry |
| AJMP | Absolute Jump |
| ANL | AND Logic |
| CJNE | Compare and Jump if Not Equal |
| CLR | Clear |
| CPL | Complement |
| DA | Decimal Adjust |
| DEC | Decrement |
| DIV | Divide |
| DJNZ | Decrement and Jump if Not Zero |
| INC | Increment |
| JB | Jump if Bit Set |
| JBC | Jump if Bit Set and Clear Bit |
| JC | Jump if Carry Set |
| JMP | Jump to Address |
| JNB | Jump if Not Bit Set |
| JNC | Jump if Carry Not Set |
| JNZ | Jump if Accumulator Not Zero |
| JZ | Jump if Accumulator Zero |
| LCALL | Long Call |
| LJMP | Long Jump |
| MOV | Move from Memory |
| MOVC | Move from Code Memory |
| MOVX | Move from Extended Memory |
| MUL | Multiply |
| NOP | No Operation |
| ORL | OR Logic |
| POP | Pop Value From Stack |
| PUSH | Push Value Onto Stack |
| RET | Return From Subroutine |
| RETI | Return From Interrupt |
| RL | Rotate Left |
| RLC | Rotate Left through Carry |
| RR | Rotate Right |
| RRC | Rotate Right through Carry |
| SETB | Set Bit |
| SJMP | Short Jump |
| SUBB | Subtract With Borrow |
| SWAP | Swap Nibbles |
| XCH | Exchange Bytes |
| XCHD | Exchange Digits |
| XRL | Exclusive OR Logic |