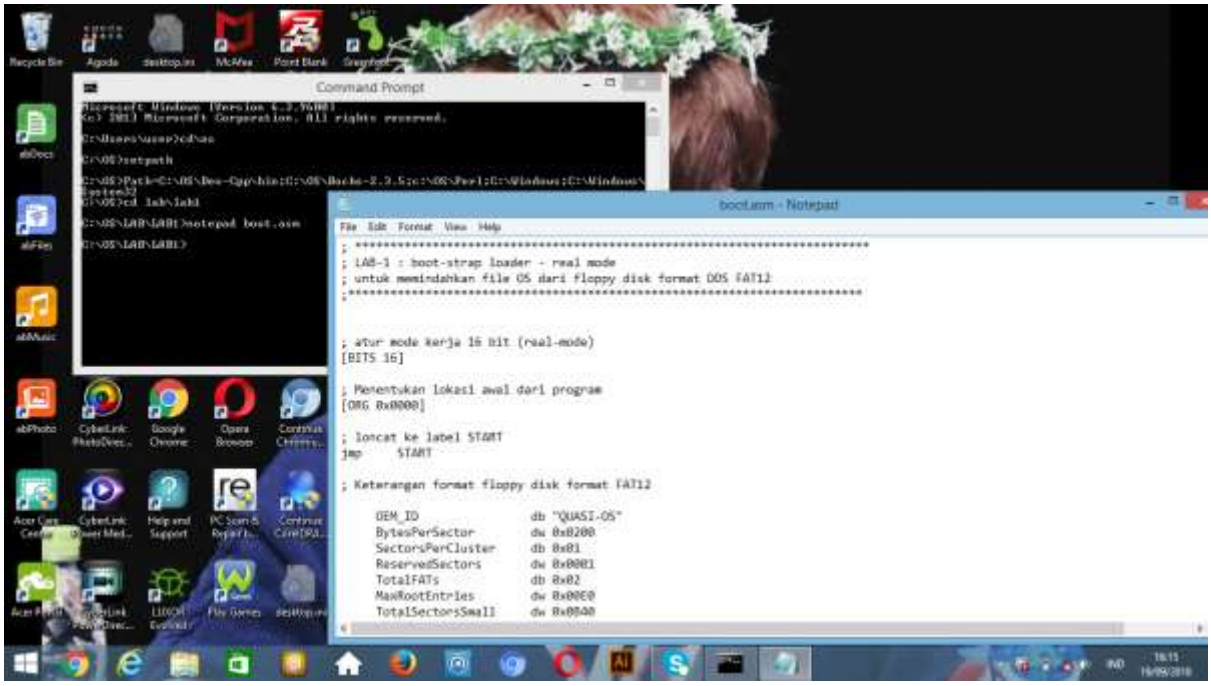
Nama : Rulla Selfiana

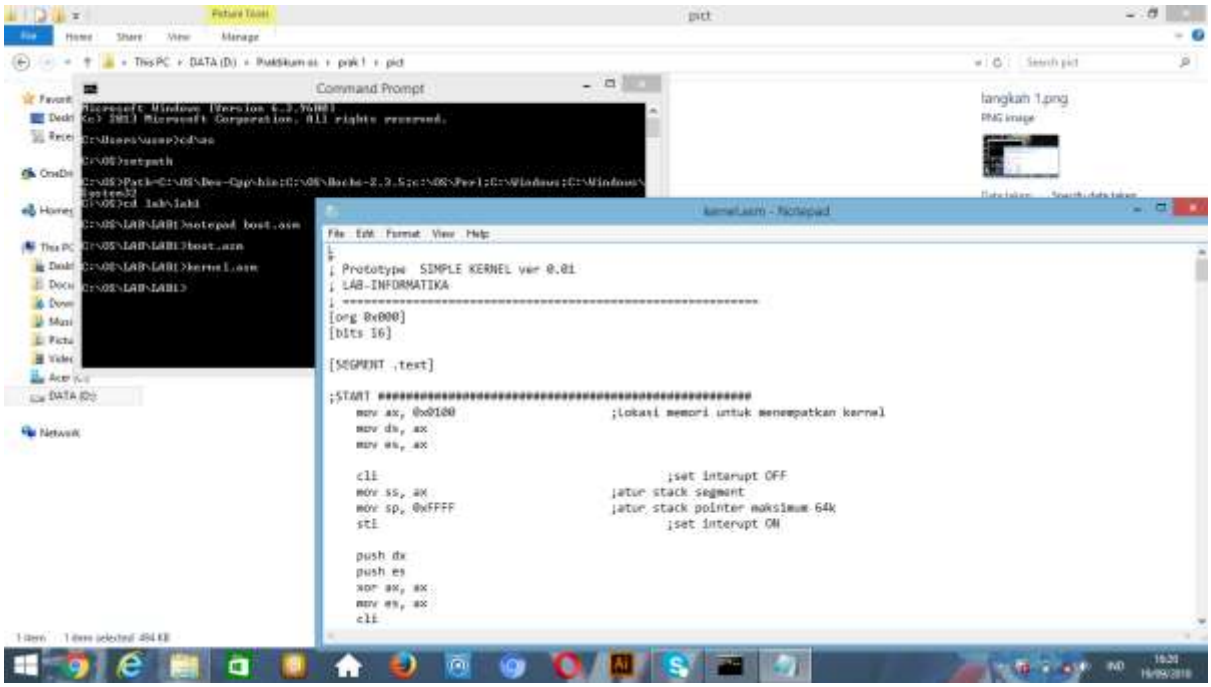
Nim : L202173005

# Praktikum

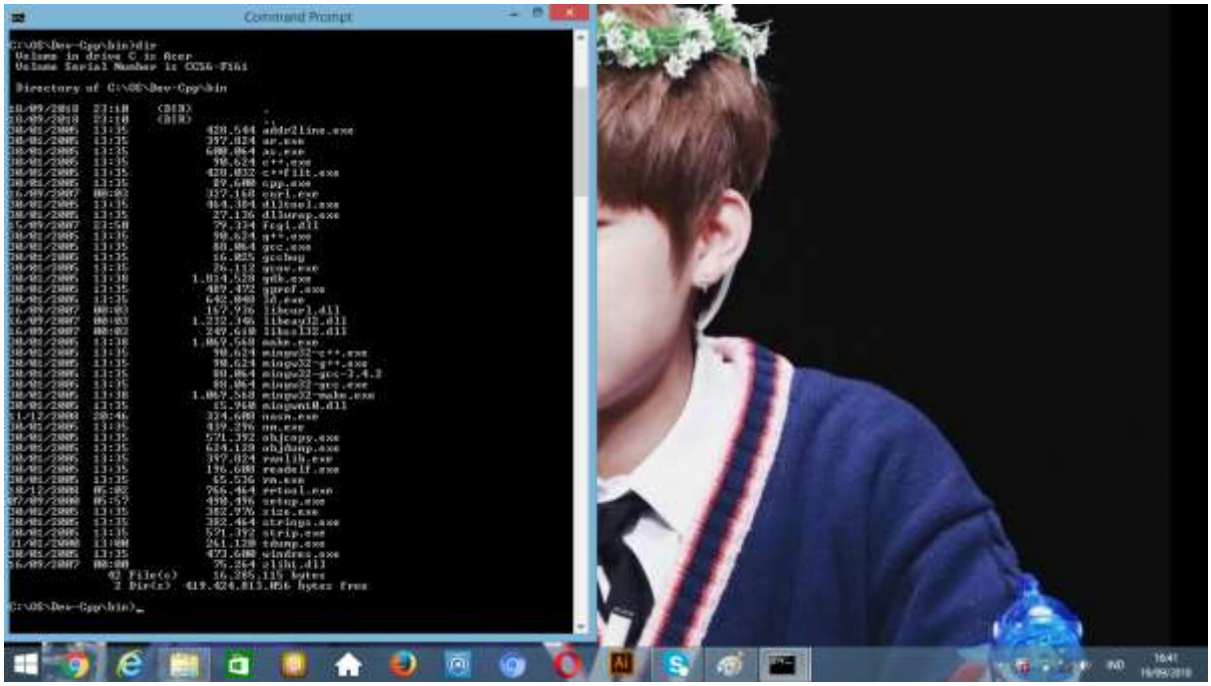
Langkah 1



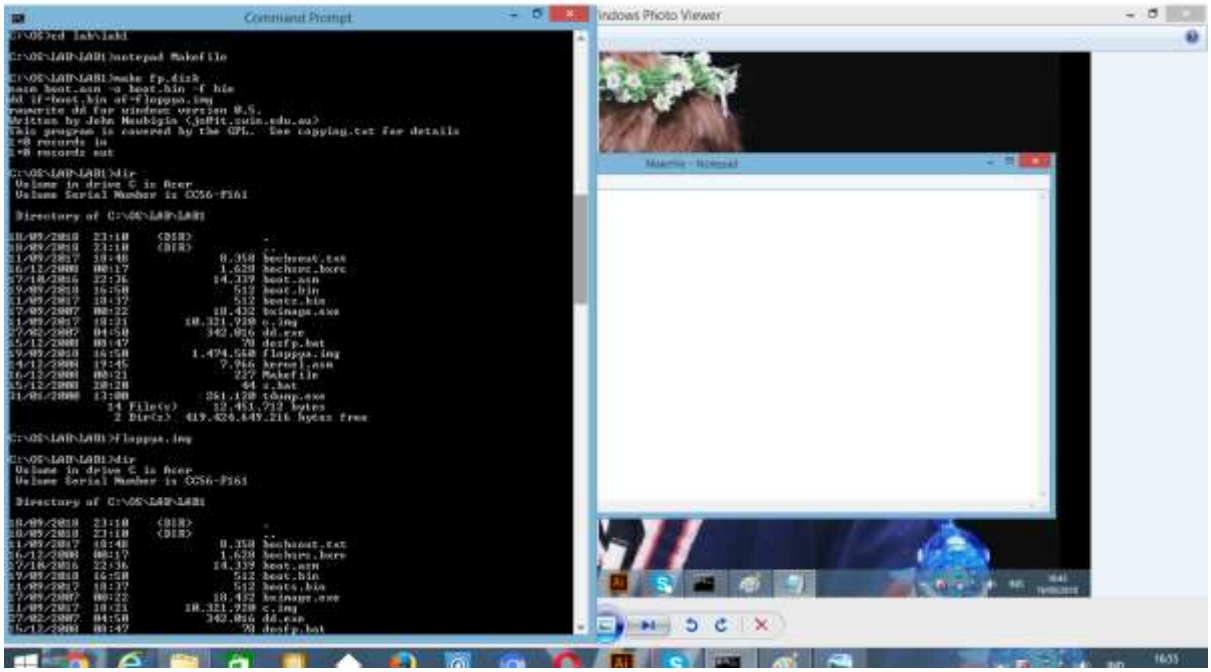

Langkah 2



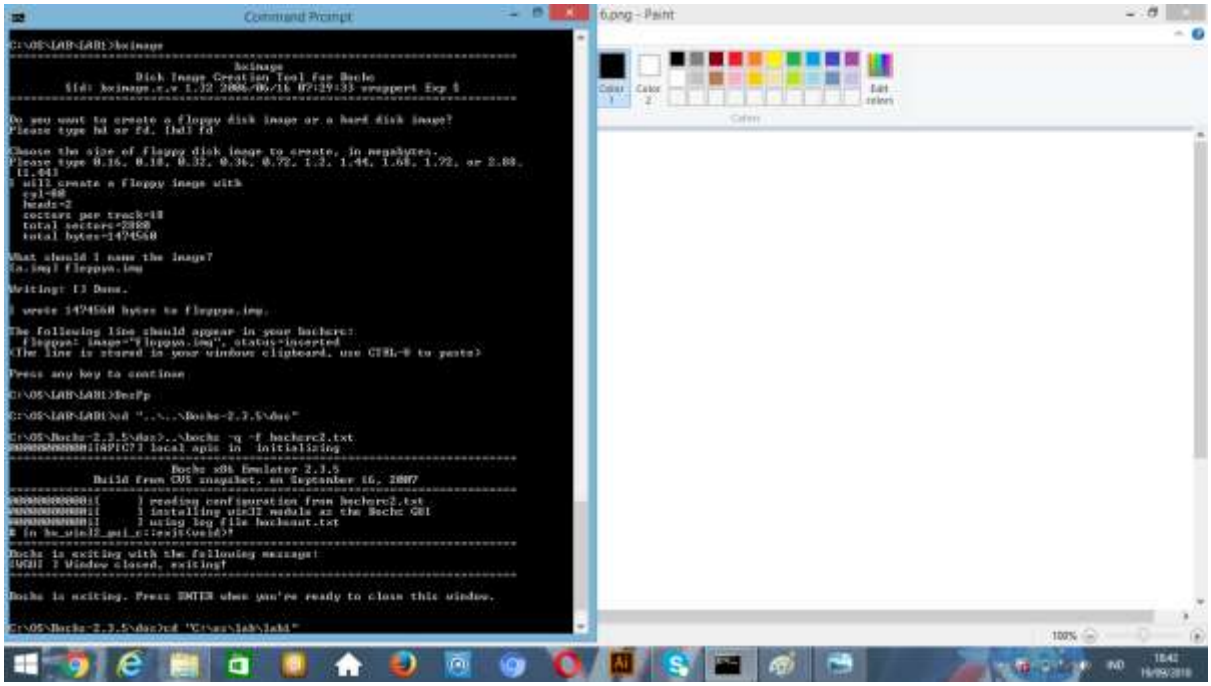
Langkah 3



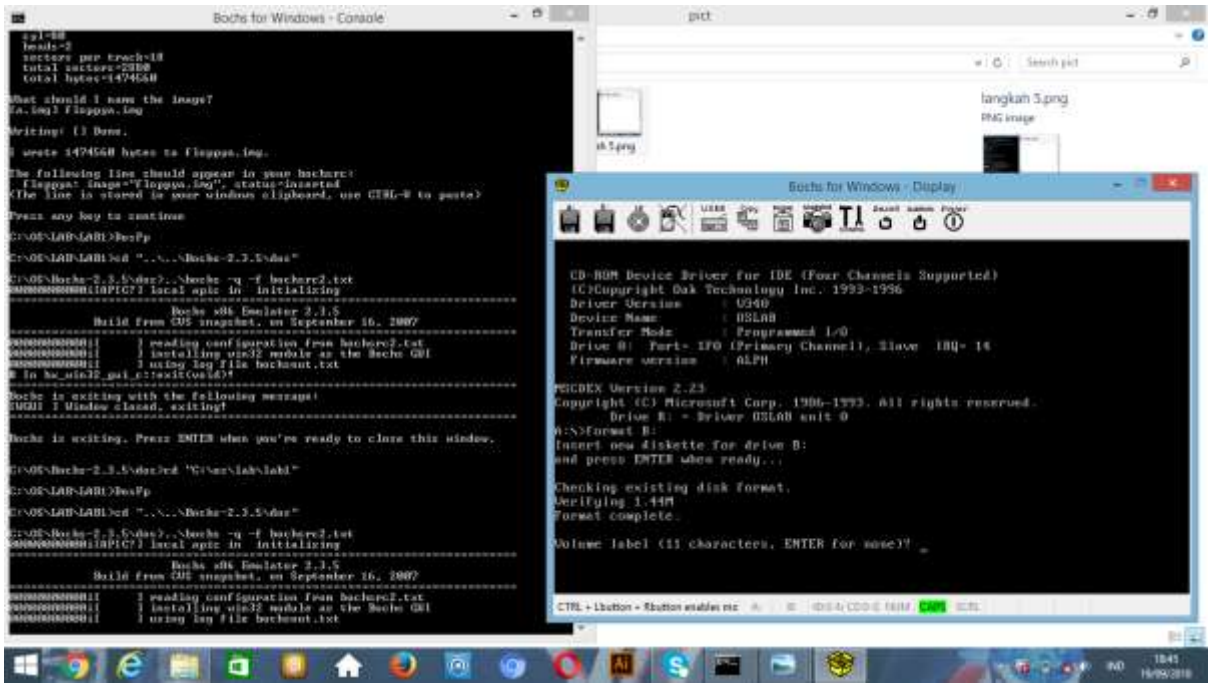
Langkah 4



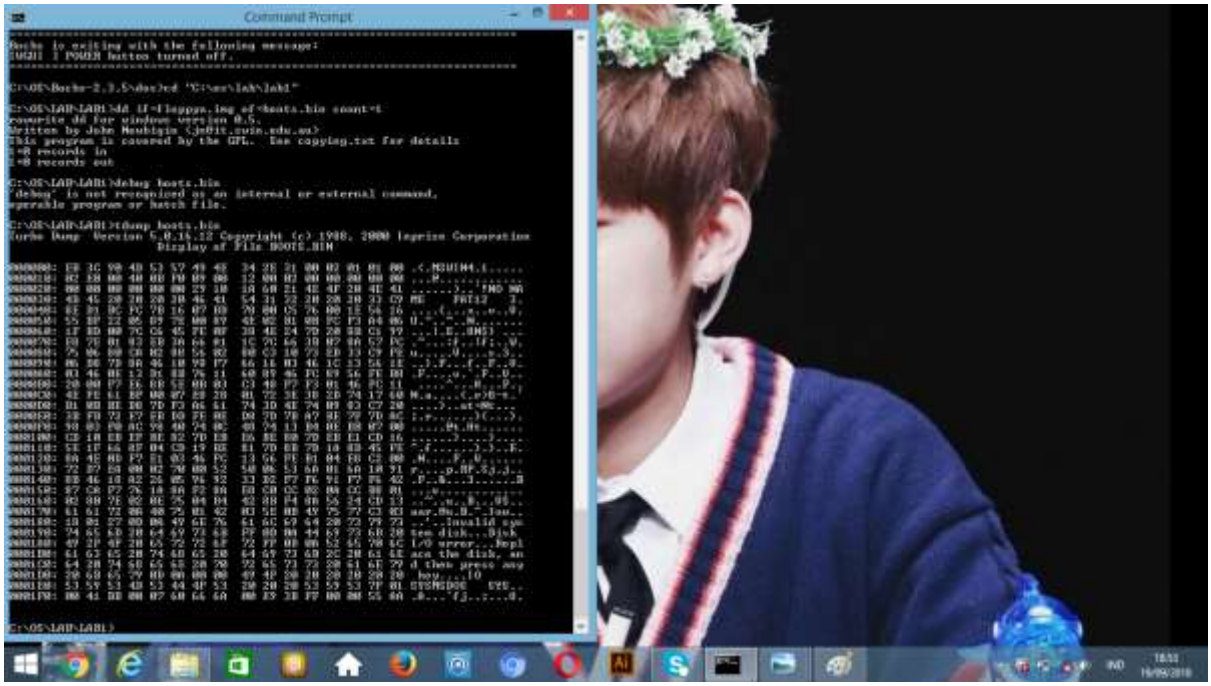
Langkah 5



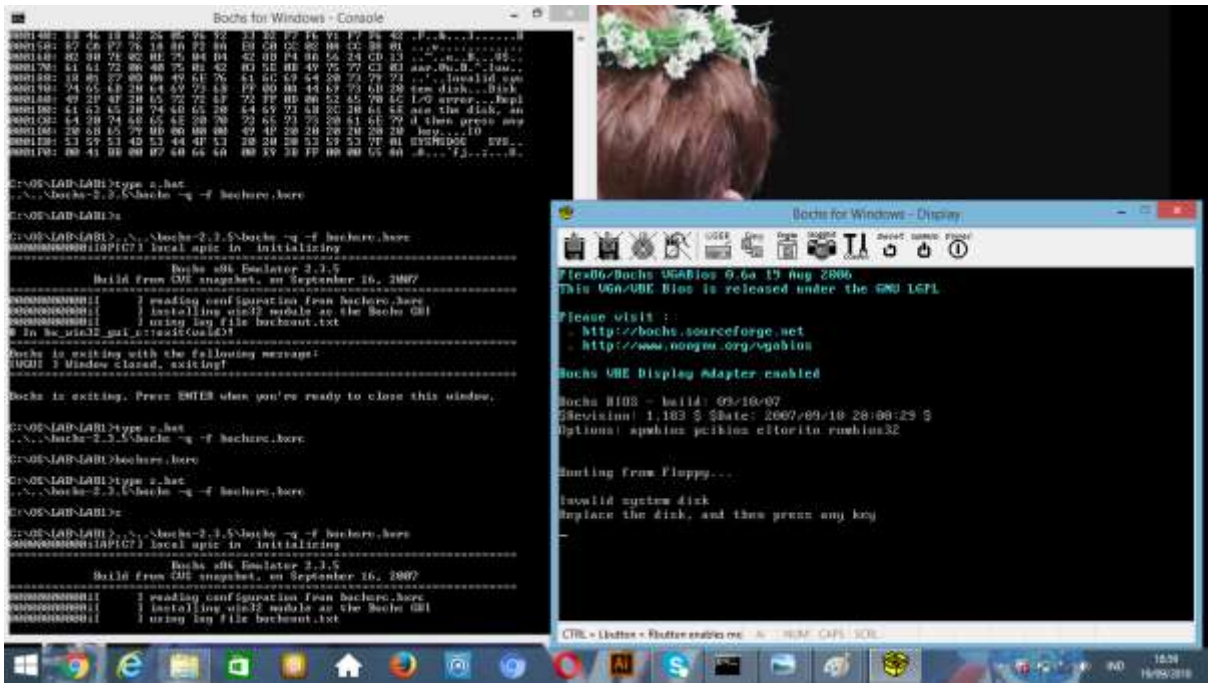

Langkah 6



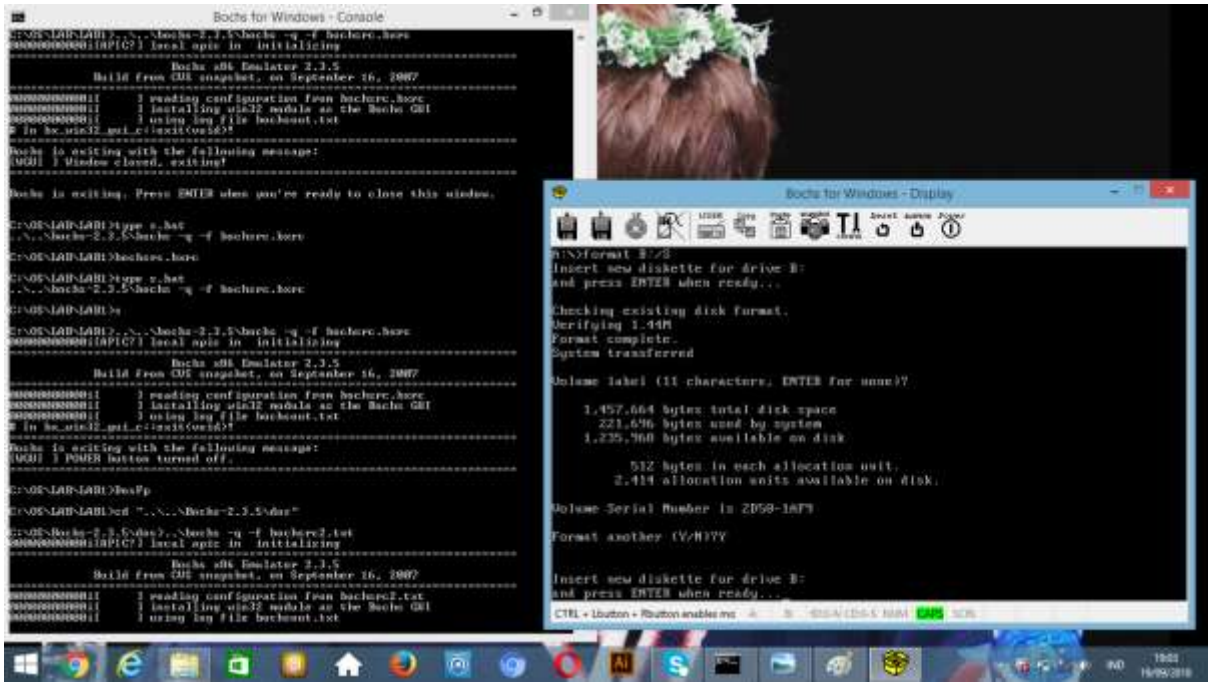

Langkah 7



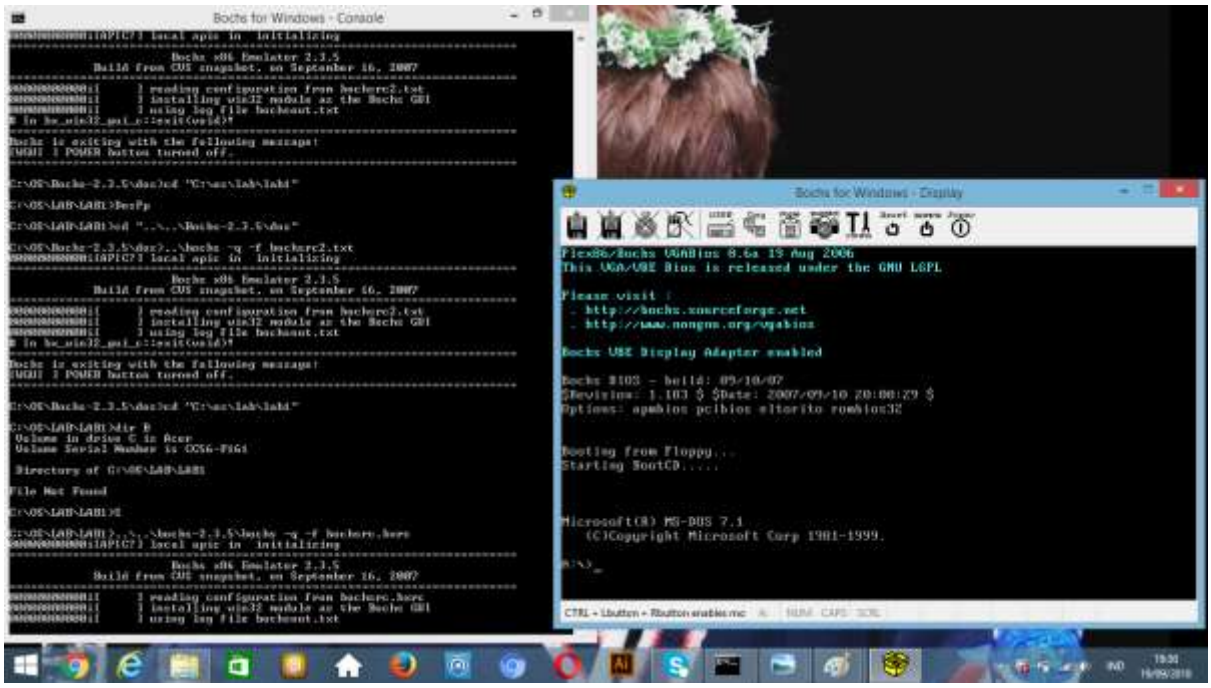

Langkah 8




Langkah 9


Bochs for Windows - Console
Bochs for Windows - Display

Langkah 10


Bochs for Windows - Console
Bochs for Windows - Display

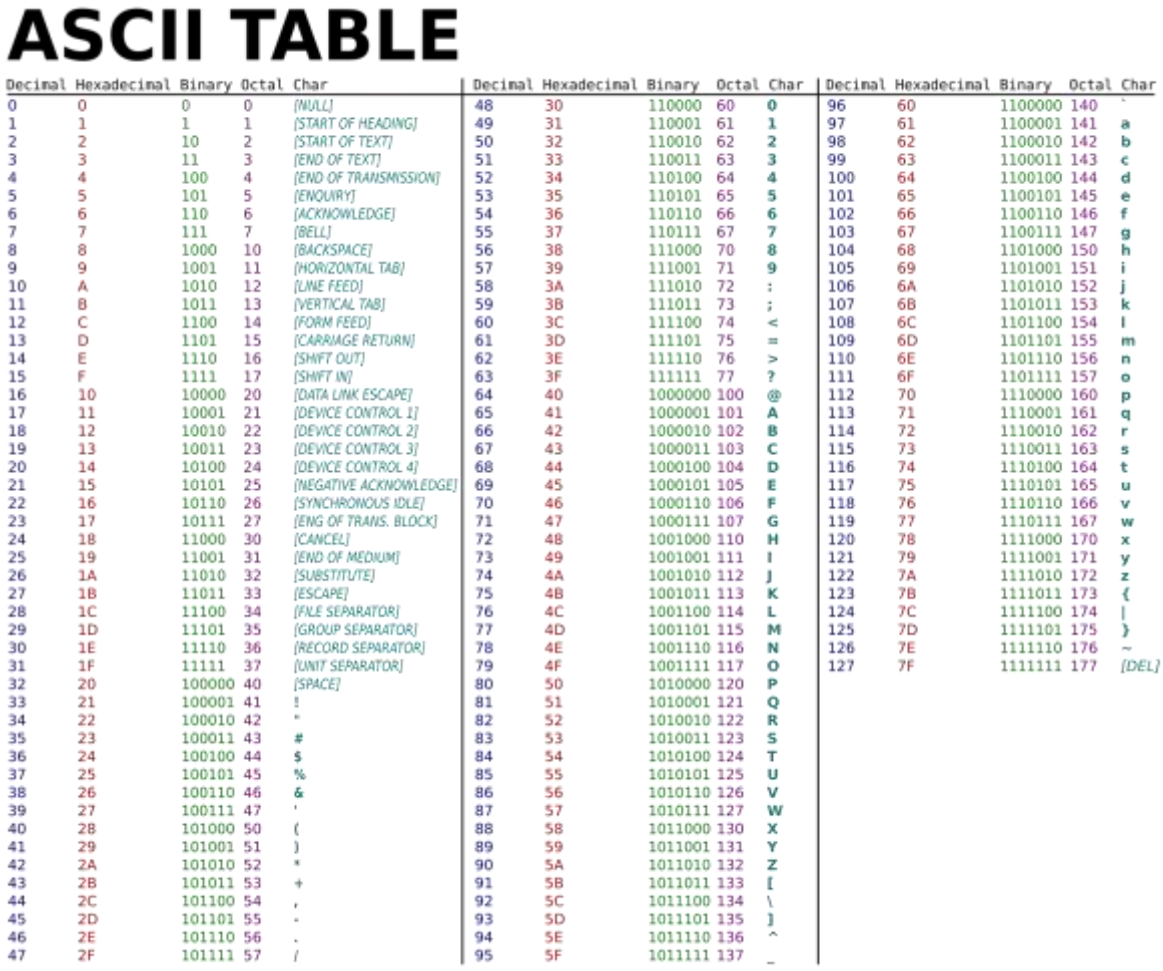
## Tugas

1. Apa itu Kode ASCII?

   **Kode Standar Amerika untuk Pertukaran Informasi** atau **ASCII** (*American Standard Code for Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|".kode ini selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan 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.

   Jumlah kode ASCII adalah 255 kode. Kode ASCII 0..127 merupakan kode ASCII untuk manipulasi teks; sedangkan kode ASCII 128..255 merupakan kode ASCII untuk manipulasi grafik.

   TABLE ASCII



# ASCII TABLE

| Decimal | Hexadecimal | Binary | Octal | Char | Decimal | Hexadecimal | Binary | Octal | Char | Decimal | Hexadecimal | Binary | Octal | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | [NULL] | 48 | 30 | 110000 | 60 | 0 | 96 | 60 | 1100000 | 140 | ` |
| 1 | 1 | 1 | 1 | [START OF HEADING] | 49 | 31 | 110001 | 61 | 1 | 97 | 61 | 1100001 | 141 | a |
| 2 | 2 | 10 | 2 | [START OF TEXT] | 50 | 32 | 110010 | 62 | 2 | 98 | 62 | 1100010 | 142 | b |
| 3 | 3 | 11 | 3 | [END OF TEXT] | 51 | 33 | 110011 | 63 | 3 | 99 | 63 | 1100011 | 143 | c |
| 4 | 4 | 100 | 4 | [END OF TRANSMISSION] | 52 | 34 | 110100 | 64 | 4 | 100 | 64 | 1100100 | 144 | d |
| 5 | 5 | 101 | 5 | [ENQUIRY] | 53 | 35 | 110101 | 65 | 5 | 101 | 65 | 1100101 | 145 | e |
| 6 | 6 | 110 | 6 | [ACKNOWLEDGE] | 54 | 36 | 110110 | 66 | 6 | 102 | 66 | 1100110 | 146 | f |
| 7 | 7 | 111 | 7 | [BELL] | 55 | 37 | 110111 | 67 | 7 | 103 | 67 | 1100111 | 147 | g |
| 8 | 8 | 1000 | 10 | [BACKSPACE] | 56 | 38 | 111000 | 70 | 8 | 104 | 68 | 1101000 | 150 | h |
| 9 | 9 | 1001 | 11 | [HORIZONTAL TAB] | 57 | 39 | 111001 | 71 | 9 | 105 | 69 | 1101001 | 151 | i |
| 10 | A | 1010 | 12 | [LINE FEED] | 58 | 3A | 111010 | 72 | : | 106 | 6A | 1101010 | 152 | j |
| 11 | B | 1011 | 13 | [VERTICAL TAB] | 59 | 3B | 111011 | 73 | ; | 107 | 6B | 1101011 | 153 | k |
| 12 | C | 1100 | 14 | [FORM FEED] | 60 | 3C | 111100 | 74 | < | 108 | 6C | 1101100 | 154 | l |
| 13 | D | 1101 | 15 | [CARRIAGE RETURN] | 61 | 3D | 111101 | 75 | = | 109 | 6D | 1101101 | 155 | m |
| 14 | E | 1110 | 16 | [SHIFT OUT] | 62 | 3E | 111110 | 76 | > | 110 | 6E | 1101110 | 156 | n |
| 15 | F | 1111 | 17 | [SHIFT IN] | 63 | 3F | 111111 | 77 | ? | 111 | 6F | 1101111 | 157 | o |
| 16 | 10 | 10000 | 20 | [DATA LINK ESCAPE] | 64 | 40 | 1000000 | 100 | @ | 112 | 70 | 1110000 | 160 | p |
| 17 | 11 | 10001 | 21 | [DEVICE CONTROL 1] | 65 | 41 | 1000001 | 101 | A | 113 | 71 | 1110001 | 161 | q |
| 18 | 12 | 10010 | 22 | [DEVICE CONTROL 2] | 66 | 42 | 1000010 | 102 | B | 114 | 72 | 1110010 | 162 | r |
| 19 | 13 | 10011 | 23 | [DEVICE CONTROL 3] | 67 | 43 | 1000011 | 103 | C | 115 | 73 | 1110011 | 163 | s |
| 20 | 14 | 10100 | 24 | [DEVICE CONTROL 4] | 68 | 44 | 1000100 | 104 | D | 116 | 74 | 1110100 | 164 | t |
| 21 | 15 | 10101 | 25 | [NEGATIVE ACKNOWLEDGE] | 69 | 45 | 1000101 | 105 | E | 117 | 75 | 1110101 | 165 | u |
| 22 | 16 | 10110 | 26 | [SYNCHRONOUS IDLE] | 70 | 46 | 1000110 | 106 | F | 118 | 76 | 1110110 | 166 | v |
| 23 | 17 | 10111 | 27 | [ENG OF TRANS. BLOCK] | 71 | 47 | 1000111 | 107 | G | 119 | 77 | 1110111 | 167 | w |
| 24 | 18 | 11000 | 30 | [CANCEL] | 72 | 48 | 1001000 | 110 | H | 120 | 78 | 1111000 | 170 | x |
| 25 | 19 | 11001 | 31 | [END OF MEDIUM] | 73 | 49 | 1001001 | 111 | I | 121 | 79 | 1111001 | 171 | y |
| 26 | 1A | 11010 | 32 | [SUBSTITUTE] | 74 | 4A | 1001010 | 112 | J | 122 | 7A | 1111010 | 172 | z |
| 27 | 1B | 11011 | 33 | [ESCAPE] | 75 | 4B | 1001011 | 113 | K | 123 | 7B | 1111011 | 173 | { |
| 28 | 1C | 11100 | 34 | [FILE SEPARATOR] | 76 | 4C | 1001100 | 114 | L | 124 | 7C | 1111100 | 174 | \| |
| 29 | 1D | 11101 | 35 | [GROUP SEPARATOR] | 77 | 4D | 1001101 | 115 | M | 125 | 7D | 1111101 | 175 | } |
| 30 | 1E | 11110 | 36 | [RECORD SEPARATOR] | 78 | 4E | 1001110 | 116 | N | 126 | 7E | 1111110 | 176 | ~ |
| 31 | 1F | 11111 | 37 | [UNIT SEPARATOR] | 79 | 4F | 1001111 | 117 | O | 127 | 7F | 1111111 | 177 | [DEL] |
| 32 | 20 | 100000 | 40 | [SPACE] | 80 | 50 | 1010000 | 120 | P | | | | | |
| 33 | 21 | 100001 | 41 | ! | 81 | 51 | 1010001 | 121 | Q | | | | | |
| 34 | 22 | 100010 | 42 | " | 82 | 52 | 1010010 | 122 | R | | | | | |
| 35 | 23 | 100011 | 43 | # | 83 | 53 | 1010011 | 123 | S | | | | | |
| 36 | 24 | 100100 | 44 | $ | 84 | 54 | 1010100 | 124 | T | | | | | |
| 37 | 25 | 100101 | 45 | % | 85 | 55 | 1010101 | 125 | U | | | | | |
| 38 | 26 | 100110 | 46 | & | 86 | 56 | 1010110 | 126 | V | | | | | |
| 39 | 27 | 100111 | 47 | ' | 87 | 57 | 1010111 | 127 | W | | | | | |
| 40 | 28 | 101000 | 50 | ( | 88 | 58 | 1011000 | 130 | X | | | | | |
| 41 | 29 | 101001 | 51 | ) | 89 | 59 | 1011001 | 131 | Y | | | | | |
| 42 | 2A | 101010 | 52 | * | 90 | 5A | 1011010 | 132 | Z | | | | | |
| 43 | 2B | 101011 | 53 | + | 91 | 5B | 1011011 | 133 | [ | | | | | |
| 44 | 2C | 101100 | 54 | , | 92 | 5C | 1011100 | 134 | \ | | | | | |
| 45 | 2D | 101101 | 55 | - | 93 | 5D | 1011101 | 135 | ] | | | | | |
| 46 | 2E | 101110 | 56 | . | 94 | 5E | 1011110 | 136 | ^ | | | | | |
| 47 | 2F | 101111 | 57 | / | 95 | 5F | 1011111 | 137 | _ | | | | | |

2. Daftar Perintah Bahasa Assembly

**Original 8086/8088 instruction set**

| Instruction | Meaning | Notes | Opcode |
|---|---|---|---|
| AAA | ASCII adjust AL after addition | used with unpacked binary coded decimal | 0x37 |
| AAD | ASCII adjust AX before division | 8086/8088 datasheet documents only base 10 version of the AAD instruction (opcode 0xD5 0x0A), but any other base will work. Later Intel's documentation has the generic form too. NEC V20 and V30 (and possibly other NEC V-series CPUs) always use base 10, and ignore the argument, causing a number of incompatibilities | 0xD5 |
| AAM | ASCII adjust AX after multiplication | Only base 10 version (Operand is 0xA) is documented, see notes for AAD | 0xD4 |
| AAS | ASCII adjust AL after subtraction | | 0x3F |
| ADC | Add with carry | `destination := destination + source + carry_flag` | 0x10…0x15, 0x80/2…0x83/2 |
| ADD | Add | (1) `r/m += r/imm;` (2) `r += m/imm;` | 0x00…0x05, |

**Original 8086/8088 instruction set**

| Instruction | Meaning | Notes | Opcode |
|---|---|---|---|
| | | | 0x80/0…0x83/0 |
| AND | Logical AND | (1) `r/m &= r/imm;` (2) `r &= m/imm;` | 0x20…0x25, 0x80/4…0x83/4 |
| CALL | Call procedure | `push eip; eip points to the instruction directly after the call` | 0x9A, 0xE8, 0xFF/2, 0xFF/3 |
| CBW | Convert byte to word | | 0x98 |
| CLC | Clear carry flag | `CF = 0;` | 0xF8 |
| CLD | Clear direction flag | `DF = 0;` | 0xFC |
| CLI | Clear interrupt flag | `IF = 0;` | 0xFA |
| CMC | Complement carry flag | | 0xF5 |
| CMP | Compare operands | | 0x38…0x3D, 0x80/7…0x83/7 |

**Original 8086/8088 instruction set**

| Instruction | Meaning | Notes | Opcode |
|---|---|---|---|
| CMPSB | Compare bytes in memory | | 0xA6 |
| CMPSW | Compare words | | 0xA7 |
| CWD | Convert word to doubleword | | 0x99 |
| DAA | Decimal adjust AL after addition | (used with packed binary coded decimal) | 0x27 |
| DAS | Decimal adjust AL after subtraction | | 0x2F |
| DEC | Decrement by 1 | | 0x48, 0xFE/1, 0xFF/1 |
| DIV | Unsigned divide | `DX:AX = DX:AX / r/m;` resulting `DX == remainder` | 0xF6/6, 0xF7/6 |
| ESC | Used with floating-point unit | | |
| HLT | Enter halt state | | 0xF4 |

**Original 8086/8088 instruction set**

| Instruction | Meaning | Notes | Opcode |
|---|---|---|---|
| IDIV | Signed divide | `DX:AX = DX:AX / r/m;` resulting `DX == remainder` | 0xF6/7, 0xF7/7 |
| IMUL | Signed multiply | (1) `DX:AX = AX * r/m;` (2) `AX = AL * r/m` | 0x69, 0x6B, 0xF6/5, 0xF7/5, 0x0FAF |
| IN | Input from port | (1) `AL = port[imm];` (2) `AL = port[DX];` (3) `AX = port[DX];` | 0xE4, 0xE5, 0xEC, 0xED |
| INC | Increment by 1 | | 0x40, 0xFE/0, 0xFF/0 |
| INT | Call to interrupt | | 0xCD |
| INTO | Call to interrupt if overflow | | 0xCE |
| IRET | Return from interrupt | | 0xCF |
| Jcc | Jump if condition | (`JA, JAE, JB, JBE, JC, JE, JG, JGE, JL, JLE, JNA, JNAE, JNB, JNBE, JNC, JNE, JNG, JNGE, JNL, JNLE, JNO, JNP, JNS, JNZ, JO, JP, JPE, JPO, JS, JZ`) | 0x70…0x7F, 0xE3, 0x0F83, 0x0F87 |

**Original 8086/8088 instruction set**

| Instruction | Meaning | Notes | Opcode |
|---|---|---|---|
| JCXZ | Jump if CX is zero | | 0xE3 |
| JMP | Jump | | 0xE9…0xEB, 0xFF/4, 0xFF/5 |
| LAHF | Load FLAGS into AH register | | 0x9F |
| LDS | Load pointer using DS | | 0xC5 |
| LEA | Load Effective Address | | 0x8D |
| LES | Load ES with pointer | | 0xC4 |
| LOCK | Assert BUS LOCK# signal | (for multiprocessing) | 0xF0 |
| LODSB | Load string byte | `if (DF==0) AL = *SI++; else AL = *SI--;` | 0xAC |
| LODSW | Load string word | `if (DF==0) AX = *SI++; else AX = *SI--;` | 0xAD |
| LOOP/LOOPx | Loop control | `(LOOPE, LOOPNE, LOOPNZ, LOOPZ) if (x && --CX) goto lbl;` | 0xE0..0xE2 |

**Original 8086/8088 instruction set**

| Instruction | Meaning | Notes | Opcode |
|---|---|---|---|
| MOV | Move | copies data from one location to another, (1) `r/m = r;` (2) `r = r/m;` | |
| MOVSB | Move byte from string to string | `if (DF==0) *(byte*)DI++ = *(byte*)SI++; else *(byte*)DI-- = *(byte*)SI--;` | 0xA4 |
| MOVSW | Move word from string to string | `if (DF==0) *(word*)DI++ = *(word*)SI++; else *(word*)DI-- = *(word*)SI--;` | 0xA5 |
| MUL | Unsigned multiply | (1) `DX:AX = AX * r/m;` (2) `AX = AL * r/m;` | |
| NEG | Two's complement negation | `r/m *= -1;` | |
| NOP | No operation | opcode equivalent to `XCHG EAX, EAX` | 0x90 |
| NOT | Negate the operand, logical NOT | `r/m ^= -1;` | |
| OR | Logical OR | (1) `r/m \|= r/imm;` (2) `r \|= m/imm;` | |

**Original 8086/8088 instruction set**

| Instruction | Meaning | Notes | Opcode |
|---|---|---|---|
| OUT | Output to port | (1) `port[imm] = AL;` (2) `port[DX] = AL;` (3) `port[DX] = AX;` | |
| POP | Pop data from stack | `r/m = *SP++;` POP CS (opcode 0x0F) works only on 8086/8088. Later CPUs use 0x0F as a prefix for newer instructions. | |
| POPF | Pop FLAGS register from stack | `FLAGS = *SP++;` | 0x9D |
| PUSH | Push data onto stack | `*--SP = r/m;` | |
| PUSHF | Push FLAGS onto stack | `*--SP = FLAGS;` | 0x9C |
| RCL | Rotate left (with carry) | | |
| RCR | Rotate right (with carry) | | |
| REPxx | Repeat MOVS/STOS/CMPS/LODS/SCAS | (`REP`, `REPE`, `REPNE`, `REPNZ`, `REPZ`) | |
| RET | Return from procedure | Not a real instruction. The assembler will translate these to a RETN or a RETF depending on the memory model of the target system. | |
| RETN | Return from near procedure | | |
| RETF | Return from far procedure | | |
| ROL | Rotate left | | |
| ROR | Rotate right | | |
| SAHF | Store AH into FLAGS | | 0x9E |
| SAL | Shift Arithmetically left (signed shift left) | (1) `r/m <<= 1;` (2) `r/m <<= CL;` | |

**Original 8086/8088 instruction set**

| Instruction | Meaning | Notes | Opcode |
|---|---|---|---|
| SAR | Shift Arithmetically right (signed shift right) | (1) `(signed) r/m >>= 1;` (2) `(signed) r/m >>= CL;` | |
| SBB | Subtraction with borrow | alternative 1-byte encoding of `SBB AL, AL` is available via undocumented SALC instruction | |
| SCASB | Compare byte string | | 0xAE |
| SCASW | Compare word string | | 0xAF |
| SHL | Shift left (unsigned shift left) | | |
| SHR | Shift right (unsigned shift right) | | |
| STC | Set carry flag | `CF = 1;` | 0xF9 |
| STD | Set direction flag | `DF = 1;` | 0xFD |
| STI | Set interrupt flag | `IF = 1;` | 0xFB |
| STOSB | Store byte in string | `if (DF==0) *ES:DI++ = AL; else *ES:DI-- = AL;` | 0xAA |
| STOSW | Store word in string | `if (DF==0) *ES:DI++ = AX; else *ES:DI-- = AX;` | 0xAB |
| SUB | Subtraction | (1) `r/m -= r/imm;` (2) `r -= m/imm;` | |
| TEST | Logical compare (AND) | (1) `r/m & r/imm;` (2) `r & m/imm;` | |
| WAIT | Wait until not busy | Waits until BUSY# pin is inactive (used with floating-point unit) | 0x9B |
| XCHG | Exchange data | `r :=: r/m;` A spinlock typically uses xchg as an atomic operation. (coma bug). | |
| XLAT | Table look-up translation | behaves like `MOV AL, [BX+AL]` | 0xD7 |
| XOR | Exclusive OR | (1) `r/m ^= r/imm;` (2) `r ^= m/imm;` | |