Micro-processor Instructions

|  |  |  |  |
| --- | --- | --- | --- |
| Mnemonics | Hex code | Instruction | Comment |
| LDA A | 0000\_0000 00 | Load register A0 of width size [15:0] | A0 = LSB of A register  Load content from memory |
| LDA A\_1 | 0000\_0001 01 | Load register A1 of width size [31 :16] of A | A1 = MSB of A register |
| LDA B | 0000\_0010 02 | Load register B of width size [15:0] | B0 = LSB of B register |
| LDA B\_1 | 0000\_0011 03 | Load register B1 of width size [31 :16] of B | B1 = MSB of B register |
| LDA C | 0000\_0100 04 | Load register C of width size [15:0] | C0 = LSB of C register |
| LDA C\_1 | 0000\_0101 05 | Load register C1 of width size [31:16] of C | C1 = MSB of C register |
| LDA D | 0000\_0110 06 | Load register D of width size [15:0] | D0 = LSB of D register |
| LDA D\_1 | 0000\_0111 07 | Load register D1 of width size [31:16] of D | D1 = MSB of D register |
|  |  |  |  |
| MOV A\_1, A | 0000\_1001 09 | Move LSB of register A content to MSB of register A | MSB means register A width size [31:16] |
| MOV B, A | 0000\_1010 0a | Move LSB of register A content to LSB of register B | LSB means register A width size [15:0] |
| MOV C, A | 0000\_1011 0b | Move LSB of register A content to LSB of register C | Move the content of register A width size [15:0] to register C width size [15:0] |
| MOV D, A | 0000\_1100 0c | Move LSB of register A content to LSB of register D |  |
|  |  |  |  |
| MOV B\_1, B | 0000\_1101 0d | Move LSB of register B content to MSB of register B | Move the content of register B width [15:0] to register B width [31:16] |
| MOV A, B | 0000\_1110 0e | Move LSB of register B content to LSB of register A |  |
| MOV C, B | 0000\_1111 0f | Move LSB of register B content to LSB of register C |  |
|  |  |  |  |
| MOV D, B | 0001\_0001 11 | Move LSB of register B content to LSB of register D |  |
| MOV C\_1, C | 0001\_0010 12 | Move LSB of register C content to MSB of register C |  |
| MOV A, C | 0001\_0011 13 | Move LSB of register C content to LSB of register A | Move the content of register C width size [15:0] to register A width size [15:0] |
| MOV B, C | 0001\_0100 14 | Move LSB of register C content to LSB of register A |  |
| MOV D, C | 0001\_0101 15 | Move LSB of register C content to LSB of register A |  |
|  |  |  |  |
| MOV A, A\_1 | 0001\_0110 16 | Move MSB of register A content to LSB of register A |  |
| MOV B, B\_1 | 0001\_0111 17 | Move MSB of register B content to LSB of register B | Move the content of register B width size [31:16] to register B width size [15:0] |
| MOV C, C\_1 | 0001\_1000 18 | Move MSB of register C content to LSB of register C |  |
| MOV D, D\_1 | 0001\_1001 19 | Move MSB of register D content to LSB of register D |  |
|  |  |  |  |
| XCHG A, B | 0001\_1010 20 | Exchange the value(content) of A with B |  |
| XCHG A, C | 0001\_1011 21 | Exchange the value(content) of A with C |  |
| XCHG A, D | 0001\_1100 22 | Exchange the value of A with D |  |
| XCHG B, C | 0001\_1101 23 | Exchange the value of B with C |  |
| XCHG B, D | 0001\_1110 24 | Exchange the value of B with D |  |
| XCHG C, D | 0001\_1111 25 | Exchange the value of C with D |  |
| STA A | 0011\_0000 30 | Store the value of register A into memory address | Mnemonics address  30 76  Store A to memory address 76 |
| STA B | 0011\_0001 31 | Store the value of register B into memory address |  |
| STA C | 0011\_0010 32 | Store the value of register C into memory address | Mnemonics address  32 88  Store C to memory address 88 |
| STA D | 0011\_0011 33 | Store the value of register D into memory address |  |
|  |  |  |  |
| CPY A, B | 0011\_0100 34 | Copy the content of A to B |  |
| CPY A, C | 0011\_0101 35 | Copy the content of A to C |  |
| CPY A, D | 0011\_0110 36 | Copy the content of A to D |  |
|  |  |  |  |
| HLT | 1111\_0000 f0 | Stop the program |  |
| Arithmetic Logical Unit Instruction |
| ADD A, B | 1000\_0000 80 | Add the value of register A and B | A + B |
| ADD A, C | 1000\_0001 81 | Add the value of register A and C | A + C |
| ADD A, D | 1000\_0010 82 | Add the value of register A and D | A + D |
| ADD B, C | 1000\_0011 83 | Add the value of register B and C | B + C |
| ADD B, D | 1000\_0100 84 | Add the value of register B and D | B + D |
| ADD C, D | 1000\_0101 85 | Add the value of register C and D | C + D |
| SUB A, B | 1000\_1000 88 | Subtract the value of register B from A | A - B |
| SUB A, C | 1000\_1001 89 | Subtract the value of register C from A | A - C |
| SUB A, D | 1000\_1010 8a | Subtract the value of register D from A | A - D |
| SUB B, C | 1000\_1011 8b | Subtract the value of register C from B | B - C |
| SUB B, D | 1000\_1100 8c | Subtract the value of register D from B | B - D |
| SUB C, D | 1000\_1101 8d | Subtract the value of register D from C | C - D |
| MUL A, B | 1001\_0000 90 | Multiply the value of register A with B | A x B |
| MUL A, C | 1001\_0001 91 | Multiply the value of register A with C | A x C |
| MUL A, D | 1001\_0010 92 | Multiply the value of register A with D | A x D |
| MUL B, C | 1001\_0011 93 | Multiply the value of register B with C | B x C |
| MUL B, D | 1001\_0100 94 | Multiply the value of register B with D | B x D |
| MUL C, D | 1001\_0101 95 | Multiply the value of register C with D | C x D |
| DIV A, B | 1001\_1000 98 | Divide the value of register A from B | A / B |
| DIV A, C | 1001\_1001 99 | Divide the value of register A from C | A / C |
| DIV A, D | 1001\_1010 9a | Divide the value of register A from D | A / D |
| DIV B, C | 1001\_1011 9b | Divide the value of register B from C | B / C |
| DIV B, D | 1001\_1100 9c | Divide the value of register B from D | B / D |
| DIV C, D | 1001\_1101 9d | Divide the value of register C from D | C / D |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| ANDBIT A, B | 1010\_0000 a0 | Bitwise AND of A and B | A & B |
| ANDBIT A, C | 1010\_0001 a1 | Bitwise AND of A and C | A & C |
| ANDBIT A, D | 1010\_0010 a2 | Bitwise AND of A and D | A & D |
| ORBIT A, B | 1010\_0100 a4 | Bitwise OR of A and B | A | B |
| ORBIT A, C | 1010\_0101 a5 | Bitwise OR of A and C | A | C |
| ORBIT A, D | 1010\_0110 a6 | Bitwise OR of A and D | A | D |
| NANDBIT A, B | 1010\_1000 a8 | Bitwise NAND of A and B | ~ (A & B) |
| NANDBIT A, C | 1010\_1001 a9 | Bitwise NAND of A and C | ~ (A & C) |
| NANDBIT A, D | 1010\_1010 aa | Bitwise NAND of A and D | ~ (A & D) |
| NORBIT A, B | 1010\_1100 ac | Bitwise NOR of A and B | ~ (A | B) |
| NORBIT A, C | 1010\_1101 ad | Bitwise NOR of A and C | ~ (A | C) |
| NORBIT A, D | 1010\_1110 ae | Bitwise NOR of A and D | ~ (A | D) |
| XORBIT A, B | 1011\_0000 b0 | Bitwise XOR of A and B | A ^ B |
| XORBIT A, C | 1011\_0001 b1 | Bitwise XOR of A and C | A ^ C |
| XORBIT A, D | 1011\_0010 b2 | Bitwise XOR of A and D | A ^ D |
| XNORBIT A, B | 1011\_0100 b4 | Bitwise XNOR of A and B | ~ (A ^ B) |
| XNORBIT A, C | 1011\_0101 b5 | Bitwise XNOR of A and C | ~ (A ^ C) |
| XNORBIT A, D | 1011\_0110 b6 | Bitwise XNOR of A and D | ~ (A ^ D) |
| NEG A | 1100\_0000 c0 | Bitwise negation of A | ! A |
| NEG B | 1100\_0001 c1 | Bitwise negation of B | ! B |
| NEG C | 1100\_0010 c2 | Bitwise negation of C | ! C |
| NEG D | 1100\_0011 c3 | Bitwise negation of D | ! D |
|  |  |  |  |
| LAND A, B | 1011\_1000 b8 | Logical AND of A and B | A && B |
| LAND A, C | 1011\_1001 b9 | Logical AND of A and C | A && C |
| LAND A, D | 1011\_1010 ba | Logical AND of A and D | A && D |
| LOR A, B | 1011\_1100 bc | Logical OR of A and B | A || B |
| LOR A, C | 1011\_1101 bd | Logical OR of A and C | A || C |
| LOR A, D | 1011\_1110 be | Logical OR of A and D | A || D |
| SHFL A | 1100\_0100 c4 | Shift left of A bitwise | E.g.: - A = 1010 to 0100 |
| SHFL B | 1100\_0101 c5 | Shift left of B bitwise | E.g.: - B = 1010 to 0100 |
| SHFL C | 1100\_0110 c6 | Shift left of C bitwise |  |
| SHFL D | 1100\_0111 c7 | Shift left of D bitwise |  |
| SHFR A | 1100\_1000 c8 | Shift right of A bitwise | E.g.: - A = 1010 to 0101 |
| SHFR B | 1100\_1001 c9 | Shift right of B bitwise | E.g.: - B = 1010 to 0101 |
| SHFR C | 1100\_1010 ca | Shift right of C bitwise |  |
| SHFR D | 1100\_1011 cb | Shift right of D bitwise |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| Special Instruction |
| JMP | 0100\_1111 4f  Not mention in your design | JUMP current address to different address | Mnemonics address  40 56  E.g.: - 40 address to 56 |
| JMP\_iF A > B | 0100\_0000 40 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  40 56  E.g.: - 40 address to 56  If condition not, then  40 41 |
| JMP\_iF A > C | 0100\_0001 41 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF A > D | 0100\_0010 42 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF B > C | 0100\_0011 43 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  50 62  E.g.: - 50 address to 62  If condition not, then  50 51 |
| JMP\_iF B > D | 0100\_0100 44 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  55 65  E.g.: - 55 address to 65  If condition not, then  55 56 |
| JMP\_iF C > D | 0100\_0101 45 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
|  |  |  |  |
| JMP\_iF A < B | 0100\_1000 48 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  40 56  E.g.: - 40 address to 56  If condition not, then  40 41 |
| JMP\_iF A < C | 0100\_1001 49 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF A < D | 0100\_1010 4a | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF B < C | 0100\_1011 4b | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  50 62  E.g.: - 50 address to 62  If condition not, then  50 51 |
| JMP\_iF B < D | 0100\_1100 4c | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  55 65  E.g.: - 55 address to 65  If condition not, then  55 56 |
| JMP\_iF C < D | 0100\_1101 4d | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
|  |  |  |  |
| JMP\_iF A >= B | 0101\_0000 50 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  40 56  E.g.: - 40 address to 56  If condition not, then  40 41 |
| JMP\_iF A >= C | 0101\_0001 51 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF A >= D | 0101\_0010 52 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF B >= C | 0101\_0011 53 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  50 62  E.g.: - 50 address to 62  If condition not, then  50 51 |
| JMP\_iF B >= D | 0101\_0100 54 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  55 65  E.g.: - 55 address to 65  If condition not, then  55 56 |
| JMP\_iF C >= D | 0101\_0101 55 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
|  |  |  |  |
| JMP\_iF A <= B | 0101\_1000 58 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  40 56  E.g.: - 40 address to 56  If condition not, then  40 41 |
| JMP\_iF A <= C | 0101\_1001 59 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF A <= D | 0101\_1010 5a | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF B <= C | 0101\_1011 5b | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  50 62  E.g.: - 50 address to 62  If condition not, then  50 51 |
| JMP\_iF B <= D | 0101\_1100 5c | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  55 65  E.g.: - 55 address to 65  If condition not, then  55 56 |
| JMP\_iF C <= D | 0101\_1101 5d | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
|  |  |  |  |
|  |  |  |  |
| JMP\_iF A == B | 0110\_0000 60 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  40 56  E.g.: - 40 address to 56  If condition not, then  40 41 |
| JMP\_iF A == C | 0110\_0001 61 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF A == D | 0110\_0010 62 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF B == C | 0110\_0011 63 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  50 62  E.g.: - 50 address to 62  If condition not, then  50 51 |
| JMP\_iF B == D | 0110\_0100 64 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  55 65  E.g.: - 55 address to 65  If condition not, then  55 56 |
| JMP\_iF C == D | 0110\_0101 65 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
|  |  |  |  |
| JMP\_iF A != B | 0110\_1000 68 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  40 56  E.g.: - 40 address to 56  If condition not, then  40 41 |
| JMP\_iF A != C | 0110\_1001 69 | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF A != D | 0110\_1010 6a | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
| JMP\_iF B != C | 0110\_1011 6b | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  50 62  E.g.: - 50 address to 62  If condition not, then  50 51 |
| JMP\_iF B != D | 0110\_1100 6c | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  55 65  E.g.: - 55 address to 65  If condition not, then  55 56 |
| JMP\_iF C != D | 0110\_1101 6d | JUMP current address to different address if condition is not met then simple increment the address | Mnemonics address  46 59  E.g.: - 46 address to 59  If condition not, then  46 47 |
|  |  |  |  |

1. Load instructions (LDA): 8
2. Move instructions (MOV): 16
3. Exchange instructions (XCHG): 6
4. Store instructions (STA): 4
5. Arithmetic Logical Unit instructions: 60
6. Special instructions (JMP, JMP\_iF): 37
7. Halt the Program (HLT): 1

Adding them up:

8 (LDA) + 16 (MOV) + 6 (EXCH) + 4 (STA) + 60 (Arithmetic Logical Unit) + 37 (Special Instructions) + 1 (Halt Program) = 132