Nama : Muhammad Yulianto

NIM : L200150057

Depending on operation they perform, all instructions are divided in several groups:

• Arithmetic Instructions

• Branch Instructions

• Data Transfer Instructions

• Logic Instructions

• Bit-oriented Instructions

The first part of each instruction, called MNEMONIC refers to the operation an instruction performs (copy, addition, logic operation etc.). Mnemonics are abbreviations of the name of operation being executed. For example:

• INC R1 – Means: Increment register R1 (increment register R1);

• LJMP LAB5 – Means: Long Jump LAB5 (long jump to the address marked as LAB5);

• JNZ LOOP – Means: Jump if Not Zero LOOP (if the number in the accumulator is not 0, jump to the address marked as LOOP);

The other part of instruction, called OPERAND is separated from mnemonic by at least one whitespace and defines data being processed by instructions. Some of the instructions have no operand, while some of them have one, two or three. If there is more than one operand in an instruction, they are separated by a comma. For example:

• RET – return from a subroutine;

• JZ TEMP – if the number in the accumulator is not 0, jump to the address marked as TEMP;

• ADD A,R3 – add R3 and accumulator;

• CJNE A,#20,LOOP – compare accumulator with 20. If they are not equal, jump to the address marked as LOOP;

**Arithmetic instructions**

Arithmetic instructions perform several basic operations such as addition, subtraction, division, multiplication etc. After execution, the result is stored in the first operand. For example:

ADD A,R1 – The result of addition (A+R1) will be stored in the accumulator.

ARITHMETIC INSTRUCTIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Mnemonic | Description | Byte | Cycle |
| ADD A,Rn | Adds the register to the accumulator | 1 | 1 |
| ADD A,direct | Adds the direct byte to the accumulator | 2 | 2 |
| ADD A,@Ri | Adds the indirect RAM to the accumulator | 1 | 2 |
| ADD A,#data | Adds the immediate data to the accumulator | 2 | 2 |
| ADDC A,Rn | Adds the register to the accumulator with a carry flag | 1 | 1 |
| ADDC A,direct | Adds the direct byte to the accumulator with a carry flag | 2 | 2 |
| ADDC A,@Ri | Adds the indirect RAM to the accumulator with a carry flag | 1 | 2 |
| ADDC A,#data | Adds the immediate data to the accumulator with a carry flag | 2 | 2 |
| SUBB A,Rn | Subtracts the register from the accumulator with a borrow | 1 | 1 |
| SUBB A,direct | Subtracts the direct byte from the accumulator with a borrow | 2 | 2 |
| SUBB A,@Ri | Subtracts the indirect RAM from the accumulator with a borrow | 1 | 2 |
| SUBB A,#data | Subtracts the immediate data from the accumulator with a borrow | 2 | 2 |
| INC A | Increments the accumulator by 1 | 1 | 1 |
| INC Rn | Increments the register by 1 | 1 | 2 |
| INC Rx | Increments the direct byte by 1 | 2 | 3 |
| INC @Ri | Increments the indirect RAM by 1 | 1 | 3 |
| DEC A | Decrements the accumulator by 1 | 1 | 1 |
| DEC Rn | Decrements the register by 1 | 1 | 1 |
| DEC Rx | Decrements the direct byte by 1 | 1 | 2 |
| DEC @Ri | Decrements the indirect RAM by 1 | 2 | 3 |
| INC DPTR | Increments the Data Pointer by 1 | 1 | 3 |
| MUL AB | Multiplies A and B | 1 | 5 |
| DIV AB | Divides A by B | 1 | 5 |
| DA A | Decimal adjustment of the accumulator according to BCD code | 1 | 1 |

**Branch Instructions**

There are two kinds of branch instructions:

Unconditional jump instructions: upon their execution a jump to a new location from where the program continues execution is executed.

Conditional jump instructions: a jump to a new program location is executed only if a specified condition is met. Otherwise, the program normally proceeds with the next instruction.

BRANCH INSTRUCTIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Mnemonic | Description | Byte | Cycle |
| ACALL addr11 | Absolute subroutine call | 2 | 6 |
| LCALL addr16 | Long subroutine call | 3 | 6 |
| RET | Returns from subroutine | 1 | 4 |
| RETI | Returns from interrupt subroutine | 1 | 4 |
| AJMP addr11 | Absolute jump | 2 | 3 |
| LJMP addr16 | Long jump | 3 | 4 |
| SJMP rel | Short jump (from –128 to +127 locations relative to the following instruction) | 2 | 3 |
| JC rel | Jump if carry flag is set. Short jump. | 2 | 3 |
| JNC rel | Jump if carry flag is not set. Short jump. | 2 | 3 |
| JB bit,rel | Jump if direct bit is set. Short jump. | 3 | 4 |
| JBC bit,rel | Jump if direct bit is set and clears bit. Short jump. | 3 | 4 |
| JMP @A+DPTR | Jump indirect relative to the DPTR | 1 | 2 |
| JZ rel | Jump if the accumulator is zero. Short jump. | 2 | 3 |
| JNZ rel | Jump if the accumulator is not zero. Short jump. | 2 | 3 |
| CJNE A,direct,rel | Compares direct byte to the accumulator and jumps if not equal. Short jump. | 3 | 4 |
| CJNE A,#data,rel | Compares immediate data to the accumulator and jumps if not equal. Short jump. | 3 | 4 |
| CJNE Rn,#data,rel | Compares immediate data to the register and jumps if not equal. Short jump. | 3 | 4 |
| CJNE @Ri,#data,rel | Compares immediate data to indirect register and jumps if not equal. Short jump. | 3 | 4 |
| DJNZ Rn,rel | Decrements register and jumps if not 0. Short jump. | 2 | 3 |
| DJNZ Rx,rel | Decrements direct byte and jump if not 0. Short jump. | 3 | 4 |
| NOP | No operation | 1 | 1 |

**Data Transfer Instructions**

Data transfer instructions move the content of one register to another. The register the content of which is moved remains unchanged. If they have the suffix “X” (MOVX), the data is exchanged with external memory.

DATA TRANSFER INSTRUCTIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Mnemonic | Description | Byte | Cycle |
| MOV A,Rn | Moves the register to the accumulator | 1 | 1 |
| MOV A,direct | Moves the direct byte to the accumulator | 2 | 2 |
| MOV A,@Ri | Moves the indirect RAM to the accumulator | 1 | 2 |
| MOV A,#data | Moves the immediate data to the accumulator | 2 | 2 |
| MOV Rn,A | Moves the accumulator to the register | 1 | 2 |
| MOV Rn,direct | Moves the direct byte to the register | 2 | 4 |
| MOV Rn,#data | Moves the immediate data to the register | 2 | 2 |
| MOV direct,A | Moves the accumulator to the direct byte | 2 | 3 |
| MOV direct,Rn | Moves the register to the direct byte | 2 | 3 |
| MOV direct,direct | Moves the direct byte to the direct byte | 3 | 4 |
| MOV direct,@Ri | Moves the indirect RAM to the direct byte | 2 | 4 |
| MOV direct,#data | Moves the immediate data to the direct byte | 3 | 3 |
| MOV @Ri,A | Moves the accumulator to the indirect RAM | 1 | 3 |
| MOV @Ri,direct | Moves the direct byte to the indirect RAM | 2 | 5 |
| MOV @Ri,#data | Moves the immediate data to the indirect RAM | 2 | 3 |
| MOV DPTR,#data | Moves a 16-bit data to the data pointer | 3 | 3 |
| MOVC A,@A+DPTR | Moves the code byte relative to the DPTR to the accumulator (address=A+DPTR) | 1 | 3 |
| MOVC A,@A+PC | Moves the code byte relative to the PC to the accumulator (address=A+PC) | 1 | 3 |
| MOVX A,@Ri | Moves the external RAM (8-bit address) to the accumulator | 1 | 3-10 |
| MOVX A,@DPTR | Moves the external RAM (16-bit address) to the accumulator | 1 | 3-10 |
| MOVX @Ri,A | Moves the accumulator to the external RAM (8-bit address) | 1 | 4-11 |
| MOVX @DPTR,A | Moves the accumulator to the external RAM (16-bit address) | 1 | 4-11 |
| PUSH direct | Pushes the direct byte onto the stack | 2 | 4 |
| POP direct | Pops the direct byte from the stack/td> | 2 | 3 |
| XCH A,Rn | Exchanges the register with the accumulator | 1 | 2 |
| XCH A,direct | Exchanges the direct byte with the accumulator | 2 | 3 |
| XCH A,@Ri | Exchanges the indirect RAM with the accumulator | 1 | 3 |
| XCHD A,@Ri | Exchanges the low-order nibble indirect RAM with the accumulator | 1 | 3 |

**Logic Instructions**

Logic instructions perform logic operations upon corresponding bits of two registers. After execution, the result is stored in the first operand.

LOGIC INSTRUCTIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Mnemonic | Description | Byte | Cycle |
| ANL A,Rn | AND register to accumulator | 1 | 1 |
| ANL A,direct | AND direct byte to accumulator | 2 | 2 |
| ANL A,@Ri | AND indirect RAM to accumulator | 1 | 2 |
| ANL A,#data | AND immediate data to accumulator | 2 | 2 |
| ANL direct,A | AND accumulator to direct byte | 2 | 3 |
| ANL direct,#data | AND immediae data to direct register | 3 | 4 |
| ORL A,Rn | OR register to accumulator | 1 | 1 |
| ORL A,direct | OR direct byte to accumulator | 2 | 2 |
| ORL A,@Ri | OR indirect RAM to accumulator | 1 | 2 |
| ORL direct,A | OR accumulator to direct byte | 2 | 3 |
| ORL direct,#data | OR immediate data to direct byte | 3 | 4 |
| XRL A,Rn | Exclusive OR register to accumulator | 1 | 1 |
| XRL A,direct | Exclusive OR direct byte to accumulator | 2 | 2 |
| XRL A,@Ri | Exclusive OR indirect RAM to accumulator | 1 | 2 |
| XRL A,#data | Exclusive OR immediate data to accumulator | 2 | 2 |
| XRL direct,A | Exclusive OR accumulator to direct byte | 2 | 3 |
| XORL direct,#data | Exclusive OR immediate data to direct byte | 3 | 4 |
| CLR A | Clears the accumulator | 1 | 1 |
| CPL A | Complements the accumulator (1=0, 0=1) | 1 | 1 |
| SWAP A | Swaps nibbles within the accumulator | 1 | 1 |
| RL A | Rotates bits in the accumulator left | 1 | 1 |
| RLC A | Rotates bits in the accumulator left through carry | 1 | 1 |
| RR A | Rotates bits in the accumulator right | 1 | 1 |
| RRC A | Rotates bits in the accumulator right through carry | 1 | 1 |

**Bit-oriented Instructions**

Similar to logic instructions, bit-oriented instructions perform logic operations. The difference is that these are performed upon single bits.

BIT-ORIENTED INSTRUCTIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Mnemonic | Description | Byte | Cycle |
| CLR C | Clears the carry flag | 1 | 1 |
| CLR bit | Clears the direct bit | 2 | 3 |
| SETB C | Sets the carry flag | 1 | 1 |
| SETB bit | Sets the direct bit | 2 | 3 |
| CPL C | Complements the carry flag | 1 | 1 |
| CPL bit | Complements the direct bit | 2 | 3 |
| ANL C,bit | AND direct bit to the carry flag | 2 | 2 |
| ANL C,/bit | AND complements of direct bit to the carry flag | 2 | 2 |
| ORL C,bit | OR direct bit to the carry flag | 2 | 2 |
| ORL C,/bit | OR complements of direct bit to the carry flag | 2 | 2 |
| MOV C,bit | Moves the direct bit to the carry flag | 2 | 2 |
| MOV bit,C | Moves the carry flag to the direct bit | 2 | 3 |

**List Instruction set of ARM**

**Arithmetic Operations**

Operations are:

|  |  |  |
| --- | --- | --- |
| – ADD | operand1 + operand2 | ; Add |
| – ADC | operand1 + operand2 + carry | ; Add with carry |
| – SUB | operand1 ‐ operand2 | ; Subtract |
| – SBC | operand1 ‐ operand2 + carry ‐1 | ; Subtract with carry |
| – RSB | operand2 ‐ operand1 | ; Reverse subtract |
| – RSC | operand2 ‐ operand1 + carry ‐ 1 | ; Reverse subtract with carry |

Syntax:

– <Operation>{<cond>}{S} Rd, Rn, Operand2

Examples

– ADD r0, r1, r2

– SUBGT r3, r3, #1

– RSBLES r4, r5, #5

**Comparisons**

The only effect of the comparisons is to update the condition flags. Thus no need to set S bit.

Operations are:

|  |  |  |
| --- | --- | --- |
| – CMP | operand1 ‐ operand2 | ; Compare |
| – CMN | operand1 + operand2 | ; Compare negative |
| – TST | operand1 AND operand2 | ; Test |
| – TEQ | operand1 EOR operand2 | ; Test equivalence |

Syntax:

– <Operation>{<cond>} Rn, Operand2

Examples:

– CMP r0, r1

– TSTEQ r2, #5

**Logical Operations**

Operations are:

|  |  |
| --- | --- |
| AND | operand1 AND operand2 |
| EOR | operand1 EOR operand2 |
| ORR | operand1 OR operand2 |
| ORN | operand1 NOR operand2 |
| BIC | operand1 AND NOT operand2 [ie bit clear] |

Syntax:

– <Operation>{<cond>}{S} Rd, Rn, Operand2

Examples:

AND r0, r1, r2

BICEQ r2, r3, #7

EORS r1,r3,r0

**Data Movement**

Operations are:

MOV operand2

MVN NOT operand2

Note that these make no use of operand1.

Syntax:

– <Operation>{<cond>}{S} Rd, Operand2

Examples:

MOV r0, r1

MOVS r2, #10

MVNEQ r1,#0