1. **The SBB computer**
   1. **Registers**

Register names consist of 3 letters, they begin with a letter indicating the size of the register.

r -> 1 byte, m -> 12 bits, n -> 4 bits

The second letter is an identifier for the register.

Third letter is its readability/writability.

w -> writeable, x -> readable and writable, i -> neither

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Size** | **Identifier** | **Description** |
| rax | Byte | A | General purpose register |
| rbx | Byte | B | Arithmetic support register |
| row | Byte | output | Halt output register |
| rii | Byte | instruction 1 | Instruction loaded from RAM |
| rxi | Byte | instruction 2 | Rest of address operand, or immediate operand for instruction |
| rsi | Byte | stack pointer | Stack memory pointer, increment and decrement operations included |
| mpx | 12 bits | program counter | increment operation included |
| mrw | 12 bits | RAM address | RAM will output what is at this address register |
| nci | 4 bits | Micro ins. counter | Used in the EEPROM address to set the control signals |

* 1. **Instructions**

Instructions are composed of two parts, (1) the operation (a name given to the instruction), and (2) the operand (which can be either a 12-bit address, 8-bit immediate value imm., or none). In the following table, registers are identified by their names given in section 1.1. The term ‘address’ is used to designate the actual value of the address, and the term ‘[address]’ is used to designate the value found at this address in memory. Conditional operations will only be executed if the required combination of flags is met (flags are carry-flag CF, zero-flag ZF, and sign-flag SF).

|  |  |  |
| --- | --- | --- |
| **Operation** | **Operand** | **Description** |
| **LDA** | **address** | [Address] -> rax, SF -> 0 |
| **ADD** | **address** | [Address] -> rbx, rax + rbx -> rax |
| **SUB** | **address** | [Address] -> rbx, rax – rbx -> rax |
| **STA** | **address** | rax -> [Address] |
| **JSR** | **address** | mpx -> stack, address -> mpx |
| **JUMP/JPNE** | **address** | Condition: not (SF and ZF)  Address -> mpx |
| **JPMC/JPEQ** | **address** | Condition: CF  Address -> mpx |
| **JMPZ/JPLT** | **address** | Condition: ZF and not SF  Address -> mpx |
| **JMPN/JPGT** | **address** | Condition: SF and not ZF  Address -> mpx |
| **AND** | **address** | [Address] -> rbx, rax & rbx -> rax |
| **OR** | **address** | [Address] -> rbx, rax | rbx -> rax |
| **CMP** | **address** | [Address] -> rbx, rax == rbx -> CF  rax <= rbx -> ZF, rax >= rbx -> SF |
| **MULTL** | **address** | [Address] -> rbx, rax \* rbx (low) -> rax |
| **MULTH** | **address** | [Address] \* rbx (high) -> rax |
| **LDI** | **imm** | imm -> rax |
| **ADD#** | **imm** | rax + imm -> rax |
| **SUB#** | **imm** | rax - imm -> rax |
| **AND#** | **imm** | rax & imm -> rax |
| **OR#** | **imm** | rax | imm -> rax |
| **LDIB** | **imm** | imm -> rbx |
| **MULTL#** | **imm** | rax \* imm (low) -> rax |
| **MULTH#** | **imm** | rax \* imm (high) -> rax |
| **PUSH#** | **imm** | imm -> stack |
| **RET#** | **imm** | stack -> mpx, imm -> rax |
| **CMP#** | **imm** | imm -> rbx, rax == rbx -> CF  rax <= rbx -> ZF, rax >= rbx -> SF |
| **HALT#** | **imm** | imm -> row, halt |
| **NOOP** |  |  |
| **OUT** |  | rax -> row |
| **INC** |  | rax + 1 -> rax |
| **DEC** |  | rax – 1 -> rax |
| **RSH** |  | rax >> 1 -> rax |
| **LSH** |  | rax << 1 -> rax |
| **TAKE** |  | rbx -> rax |
| **PUSH** |  | rax -> stack |
| **POP** |  | stack -> rax |
| **MOVE** |  | rax -> rbx |
| **RET** |  | stack -> mpx |
| **ADDC** |  | Condition: CF  rax + 1 -> rax |
| **NOT** |  | ~rax -> rax |
| **REFR** |  | refresh peripherals |
| **SUBC** |  | Condition: CF  rax – 1 -> rax |
| **HALT** |  | rax -> row, halt |