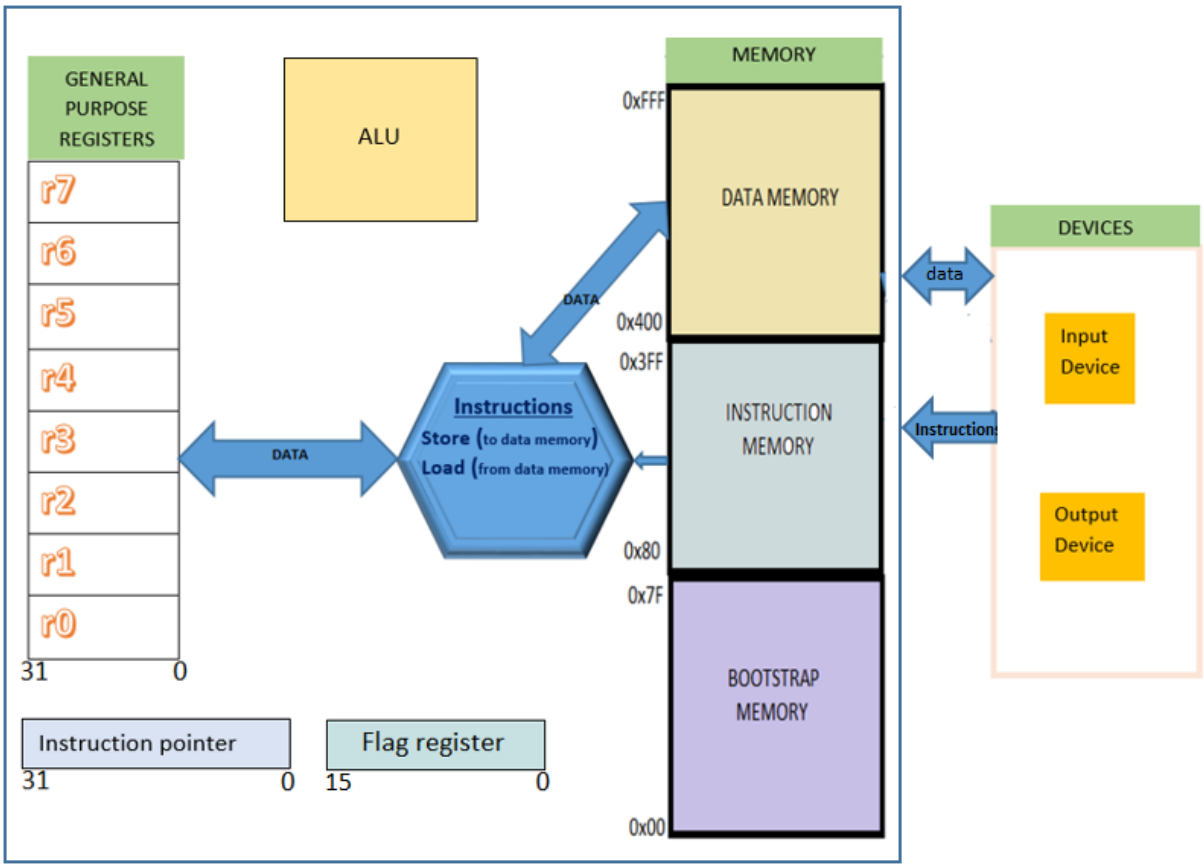
***CPU ARCHITECTURE :***




## ***Specifications***

**Main memory:**

1. *Bootstrap memory* : 0-127 : 128 bytes
2. *Instruction memory* : 128 - 1023 : 896 bytes
3. *Data memory :* 1024 - 4095 : 3.072 KB

**Description**: Memory is byte addressable (data type-uint8_t), and word size is 32 bits/4 bytes.

**Registers:**

*General purpose registers*: r0-r7 : all 4 word i.e. uint32_t
*Number of registers* : 8 registers

**Special Purpose Registers:**

*Flag_register* : 16 bit register that updates in case of Memory overflow with a value to set the 12 bit.
*Instruction Pointer*: 32 bit register which holds the address of the next instruction to be executed. It is incremented by 4 bytes after every instruction and thus points to the next instruction.(Since instruction is 32 bits wide)

**Arithmetic Logical Unit:**

Performs logical operation on the data and provides it to the registers(To be implemented in future)

## ***Operations***

1. **Load Operation**

   **Instruction:** lw register, (offset)memory address

   **Method Implemented:** int loadreg(char* reg1,uint32_t* reg_num, int memory_addr );

   - Method Name: loadreg
   - Return type: int
   - Register name: reg1 (r0-r7)
   - Pointer to the Register: reg_num (pointer to reg1)
   - Memory Address: memory_addr

   **Method Description:** After taking input from user, 32-bit register is loaded with values that are present in 4 consecutive 8 bit/byte addressable memory locations. The final memory location is the result of addition of offset address as well as memory address.

2. **Store Operation**

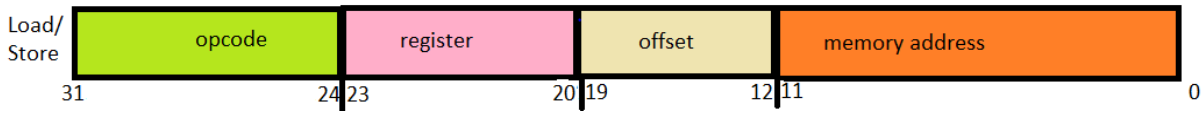
   **Instruction:** sw register,(offset)memory address

   **Method Implemented:** int storereg(char*reg1,uint32_t* reg_num, int memory_addr );

   - Method Name: storereg
   - Return type: int
   - Register name: reg1 (r0-r7)
   - Pointer to the Register: reg_num (pointer to reg1)
   - Memory Address: memory_addr

   **Method Description:** After taking input from user, values are loaded from the register into 4 consecutive 8 bit/byte addressable memory locations. The final memory location is the result of addition of offset address as well as memory address.

## ***Instruction format***




**Instruction size** : 4 bytes (opcode+register+offset+memory address)

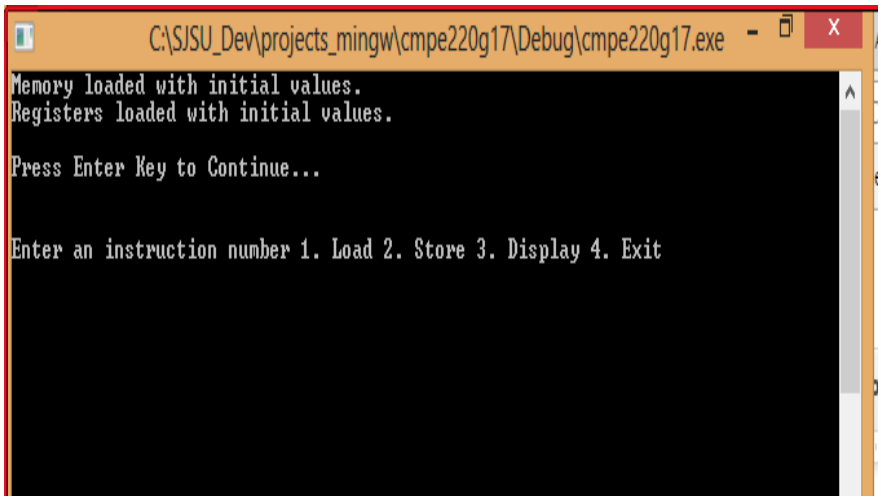
The CPU implements a 32-bit instruction set architecture with Little-Endian format. Each memory related instruction (Load/Store) is designed as below with the following fields:

- **Opcode**: 8 bits (0x00→Load, 0x01→Store) This field specifies the type of the operation being performed in the instruction. The MSB byte of the instruction represent the opcode.
- **Register**: 4 bits (0x00-0x07 → r0-r7) This field represents the number corresponding to the actual register which is being used in the Load/Store operation. This field occupies the MSB nibble next to the opcode.

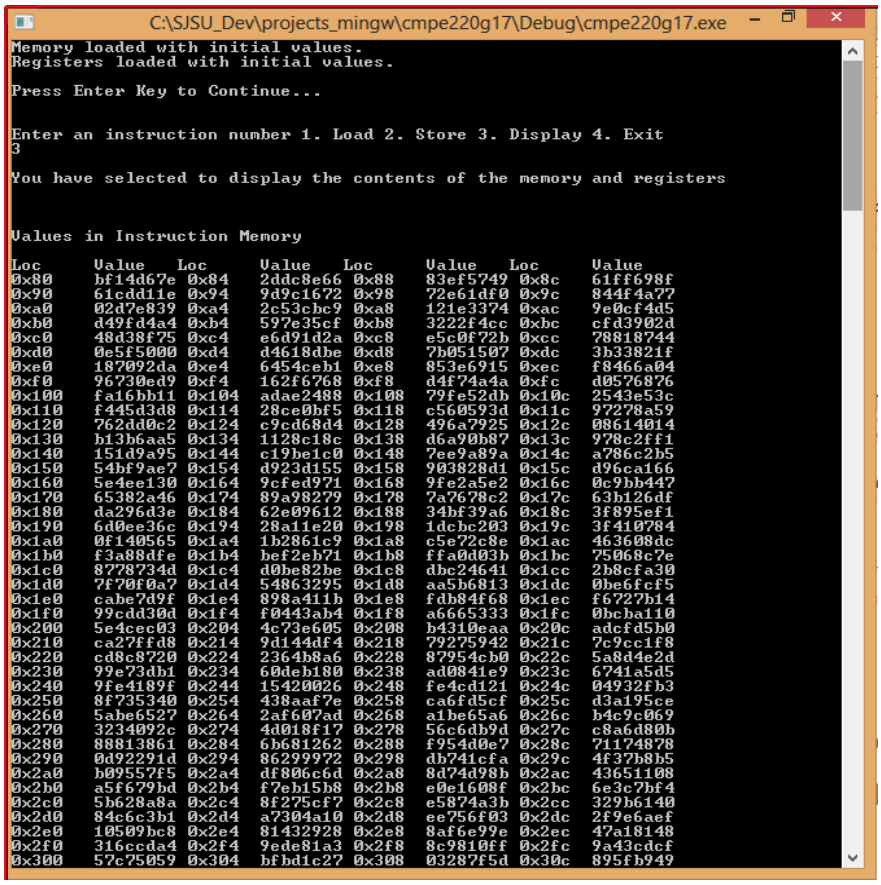
- **Offset**: 8 bits (0x00-0xff) This field specifies the offset amount of the memory address to be accessed. This takes the byte next to the register field in the instruction.
- **Memory**: 12 bits (0x00-0xfff) This field represents the address of memory from/to which a memory access should be performed. The width of this LSB field is chosen such that the instruction allows to access the memory available (i.e 4096 unique memory locations).

## ***Screen Shots***

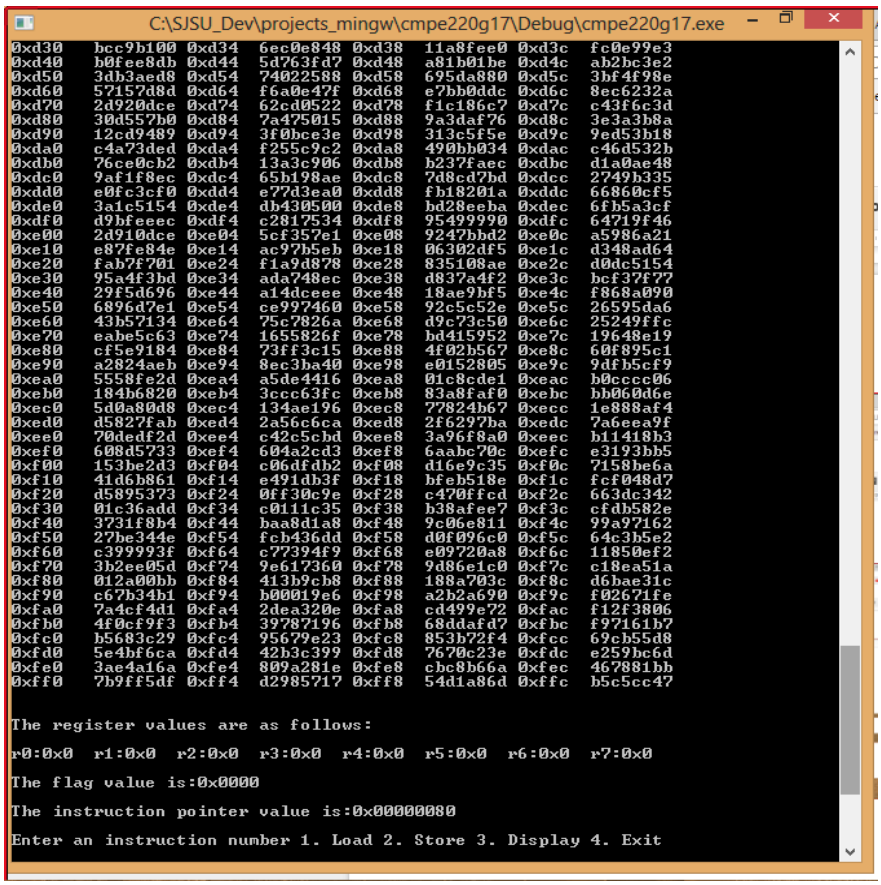
### 1) ***Initial Processor Start :***



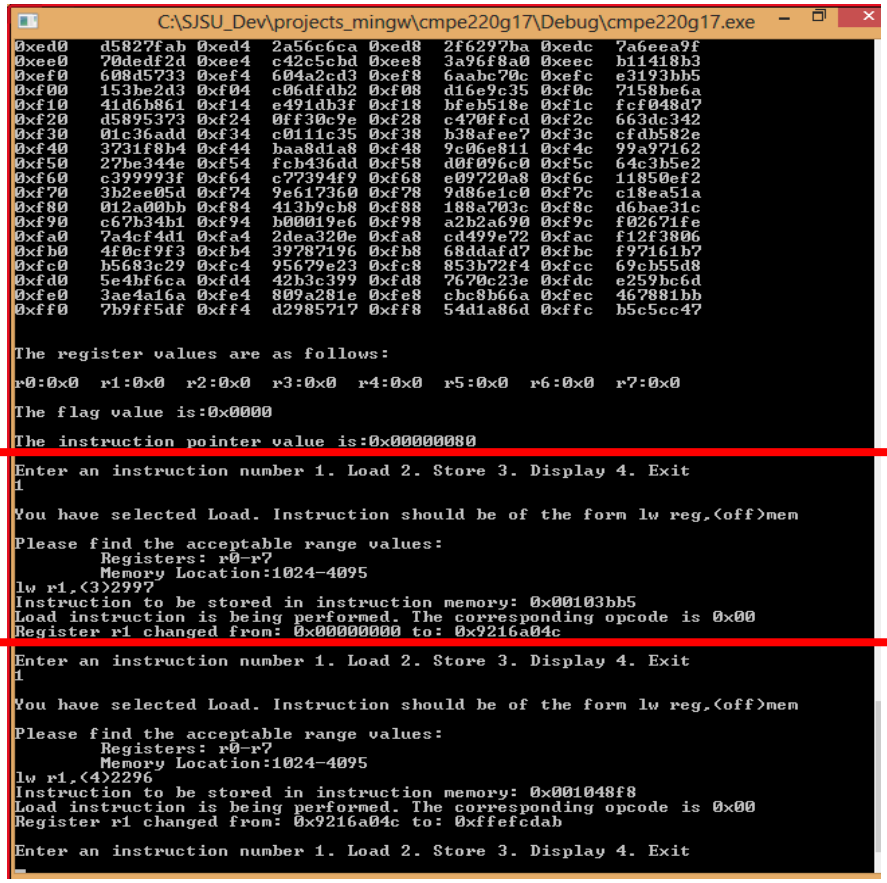

### 2.1) ***Display of Memory and Registers:***



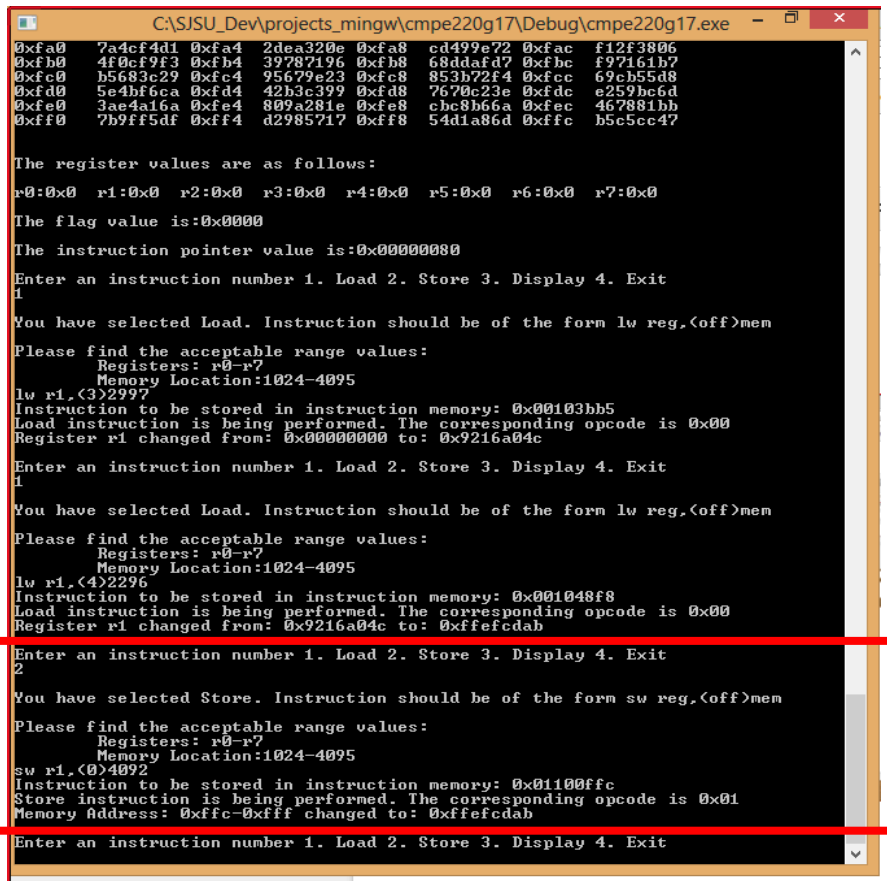
## 2.2) Display Memory and Registers (Contd..)



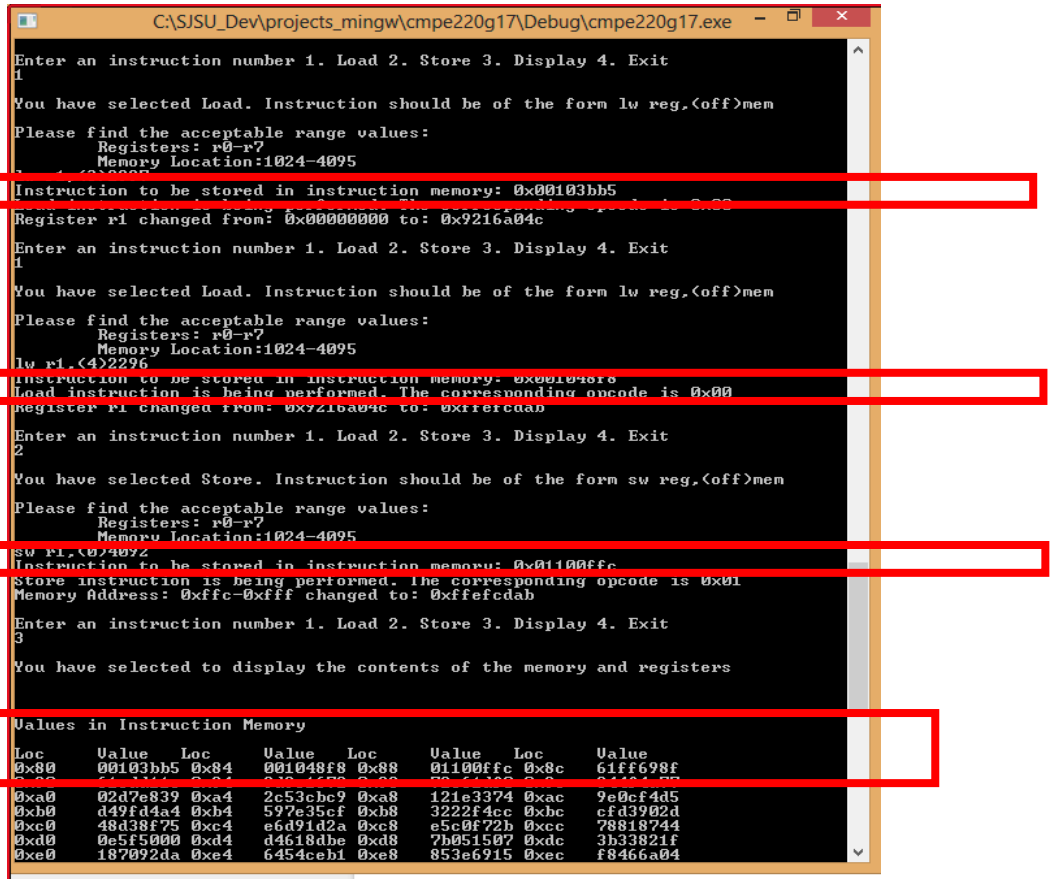

## 3) Load Instruction:



## 4) Store Instruction:




## 5) Instruction Pointer Increment:



## 6) ***Results after Load/Store:***