**CPU Architecture Diagram and Description**

# **Diagram:**

Memory[64000]

General purpose registers

regFile[8][32]

Bootstrap [0-512]

CU

ALU

$1

Text [512-660]

$2

Data [661-1023]

$3

$4

$5

Heap [1024-1535]

$6

Stack [1535-63999]

$7

$8

Flags[32]

Special purpose registers

Control\_Flag

Overflow\_Flag

Sign\_Flag

Zero\_Flag

Carry\_Flag

Parity\_Flag

Program counter (PC[16])

Memory Data (memData[32])

Memory Address (memAddr[32])

Instruction Register (Instreg[32])

# **Description:**

* **Bootstrap** – The system bootstrap or boot-up does functionalities like bringing the OS into the memory and making the processor to execute it, initializing the kernel data structures, device initialization, creating user processes and transferring control to one of them.

**In our architecture we have stored it at address starting from 0 to 512 bits.**

**Signature in code-**

#define BOOT\_ADDR 0

#define BOOT\_SECTOR 0

* **Main Memory** – The main memory is directly accessible by the CPU to store the active program and data. It is also referred to as RAM. Program is copied into the main memory from the secondary storage. Moving the instructions and data in and out of the main memory is faster since the CPU is intimately connected the memory.

**In our architecture design the main memory is of** **size is 64kb**

**Signature in code-**

char \*memory[64000]: is allocated separately using malloc since 64kb cannot be allocated in one time.

Text Segment- 512-660 bits, Data Segment- 661-1023 bits, Heap Segment- 1024-1535 bits ,Stack Segment- 1536-63999 bits,

* **General Purpose Registers**- These registers are used to store transient data required by the program. It doesn’t store any specific type of information but stores operands as well as addresses used while executing the program.

**In our architecture design we have used 8 registers ($1 to $8)**

**Signature in code-**

char regFile[8][32]

* **Special Purpose Registers-** Each special purpose register has a specific control or data handling task to be done.

1. **Instruction Register(IR)-** It holds the instruction that is currently being executed.
2. **Memory Data Register (MDR)-** It used for storing the piece of data that is fetched from the memory.
3. **Memory Address Register (MAR)-** It is used to store the address of next piece of memory to be fetched.
4. **Program Counter (PC)-** It stores the location of next instruction to be executed from the memory. It gets incremented automatically.

**Signature in code-**

1. char PC[16] for program counter
2. char memAddr[32] for memory address register
3. char memData[32] for memory data register
4. char instReg[32] for instruction register

* **Flags-** Flag register is the status register that contains the current state of the processor. We have used overflow and zero flags.

**Signature in code-**

char flags[32] for flags

1. **Overflow flag-** It indicates that the signed result of any operation is too large to fit in register width.

**Signature in code-**

#define OVERFLOW\_FLAG 1

1. **parity flag** – It indicates the number of set bits are odd or even in the binary representation of the result.
2. **Zero Flag- It** indicates that result of logical or arithmetic operation was 0.

**Signature in code-**

#define ZERO\_FLAG 0

1. **Carry flag-** It indicates if a carry or borrow is generated out of the MSB bit.
2. **Sign flag-** It is set to one when result is negative and 0 when result is positive.

* **Instruction type (Immediate Instruction)-**

Opcode base destination offset

Eg: lw $8 0x60($10)