**Cs-220 Project**

**Name - Anshul Kumar, Deepanshu**

PDS1:

As per the given problem statement, we have a total of 32 registers available in the CSE-BUBBLE processor. Following are the registers and their usage protocol:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Register number** | **Usage** | **Preserved on call?** |
| $zero | 0 | The constant value 0 | n.a. |
| $v0–$v1 | 2–3 | Values for results and expression evaluation | no |
| $a0–$a3 | 4–7 | Arguments | no |
| $t0–$t7 | 8–15 | Temporaries | no |
| $s0–$s7 | 16–23 | Saved | yes |
| $t8–$t9 | 24–25 | More temporaries | no |
| $gp | 28 | Global pointer | yes |
| $sp | 29 | Stack pointer | yes |
| $fp | 30 | Frame pointer | yes |
| $ra | 31 | Return address | yes |

The usage protocol for the registers is as follows:

Registers R0 and R24-R27 are dedicated and cannot be used for general purpose operations.

Registers R1-R23 are general purpose and can be used for arithmetic, logical, and data transfer operations.

Registers R1, R28, and R29 are reserved for specific purposes (assembler's temporary, global pointer, and stack pointer, respectively) and should not be used for general purpose operations.

Registers R4-R15 are callee-saved registers, which means they are preserved across function calls. If a function uses these registers,

it must save their contents before modifying them and restore them before returning.

Registers R16-R23 are caller-saved registers, which means their contents are not preserved across function calls. If a function uses these

registers, it does not need to save their contents before modifying them, but it must restore their original values before returning.

Registers R0-R3 are temporary registers that can be used for any purpose, but their contents may be modified by a function call and

therefore should not be relied upon to preserve their values across function calls.

**PDS2:**

The instruction size for the CSE-BUBBLE processor is 32 bits. Therefore, the instruction memory size will depend on the number of instructions we plan to store in it.

Let's assume that we have a total of 512 instructions. In that case, the instruction memory size will be 2048 bytes.

For the data memory, we will use veda memory. The veda memory has a capacity of 64KB. We will assume that the data memory size for our processor will be 2KB.

Therefore, we will use the first 2KB of the veda memory for data storage. The remaining 462KB will be reserved for instruction storage.

**PDS3:**

Here is the encoding methodology for each instruction type:

Here opcode refers to 6 MSB of instruction.

The opcode for all instructions will be

opcode value 0 means instruction is used for add

opcode value 1 means instruction is used for sub

opcode value 2 means instruction is used for addu

opcode value 3 means instruction is used for subu

opcode value 4 means instruction is used for addi

opcode value 5 means instruction is used for addiu

opcode value 6 means instruction is used for and

opcode value 7 means instruction is used for or

opcode value 8 means instruction is used for andi

opcode value 9 means instruction is used for ori

opcode value 10 means instruction is used for sll

opcode value 11 means instruction is used for srl

opcode value 12 means instruction is used for lw

opcode value 13 means instruction is used for sw

opcode value 14 means instruction is used for beq

opcode value 15 means instruction is used for bne

opcode value 16 means instruction is used for bgt

opcode value 17 means instruction is used for bgte

opcode value 18 means instruction is used for ble

opcode value 19 means instruction is used for bleq

opcode value 20 means instruction is used for j

opcode value 21 means instruction is used for jr

opcode value 22 means instruction is used for jal

opcode value 23 means instruction is used for li

opcode value 24 means instruction is used for slt