CS220 Assignment 7 Documentation
CSE-BUBBLE

Devansh Kumar Jha (200318)
Shivang Pandey (200941)

April 16, 2023

1 Milestone 1 - Registers and Usage Protocol

1.1 DEFINING THE GENERAL PURPOSE AND OTHER REGISTERS/WIRES INSIDE THE PROCESSOR

A total 32 registers inside the processor each of 32 bits have been defined. Most of these registers have functionality similar to MIPS-32 ISA. The registers have been divided into system controlled and user controlled registers.
Registers 0-5 : System controlled registers PC, EPC, Cause, BadVAddr, Status, IR
Registers 6-31 : User controlled registers r0, at, v0-v1, a0-a3, gp, sp, ra, t0-t6, s0-s7

The specific functionality and usage protocol is defined as follows -
Register 0 - PC - Program Counter (Denotes the next instruction to be fetched)
Register 1 - EPC - Exception Program Counter will denote the location of interrupt handler in case of exception.
Register 2 - Cause - This will denote the source of exception which has caused an interrupt.
Register 3 - BadVAddr - In case of branching instructions if wrong instruction is loaded then we need to wait for a clock cycle
Register 4 - Status - This signifies for how much time the processor has been waiting due to a conditional branching instruction
Register 5 - IR - It will Store the Current Instruction which is being executed
Register 6 - r0 - This register will be hardwired to 0 at all times
Register 7 - at - This register will be used by the Assembler time to time to implement Pseudo Instructions
Register (8-9) - (v0-v1) - This will be used for system calls and system instructions by the user
Register (10-13) - (a0-a3) - Will be used to provide arguments for function or system calls by the user
Register 14 - gp - Global Pointer - Will be pointing to the start of the global area,
can be used to point the starting address of heap in data memory
Register 15 - sp - Stack Pointer - Will denote the starting location of stack memory in data memory
Register 16 - ra - Return Address - Will Store the address of the instruction where we have to return after function exits
Register (17-23) - (t0-t6) - Temprorary Registers - Will be used to store values just required temprorarily

Register (24-31) - (s0-s7) - Stored Registers - Will be used to store values required over multiple functions or module

The differences between these defined set of registers and MIPS-32 ISA is that there are no floating point registers in CSE-BUBBLE, and since multiplication and division is not defined in CSE-BUBBLE the HI and LO registers are also not required. Also to keep the number of registers limited to 32, we have only kept 7 temporary and 8 stored register.

2 Milestone 2 - Instruction and Data Memory

Size of both the data and instruction memory has been set to **32x256 bits.**

3 Milestone 3 - Instruction Encoding

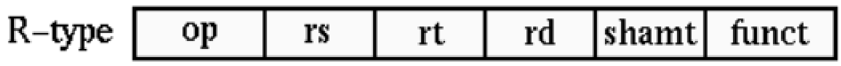
The instructions are encoded within the following categories broadly -

R type Instructions (32 bits) = Opcode (6 bits)[31:26] + Argument 1 (5 bits)[25:21] + Argument 2 (5 bits)[20:16] + Destination (Argument 3) (5 bits)[15:11] + Shift Amount (5 bits)[10:6] + Function (6 bits)[5:0]

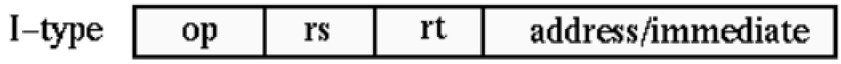
I type Instructions (32 bits) = Opcode (6 bits)[31:26] + Argument 1 (5 bits)[25:21] + Destination (Argument 3) (5 bits)[20:16] + Constant (Argument 2) (16 bits)[15:0]

J type Instructions (32 bits) = Opcode (6 bits)[31:26] + Constant (Argument 1) (26 bits)[25:0]

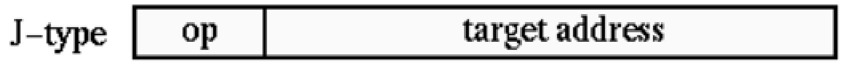
The instruction layout has been set the same as MIPS-32 ISA including the layouts for R-, J-, and I- type instructions.



Arithmetic instruction format



Transfer, branch, immediate.



Jump instruction

The ISA CSE-BUBBLE implements the following 25 Instructions which are explained below, along with the opcode and function. We have also included some special system instructions apart from the ones mentioned in the question.

3.0.1 Arithemetic Instructions

1: add r0, r1, r2 - R type - Opcode: 0 Function: 0
2: sub r0, r1, r2 - R type - Opcode: 0 Function: 1
3: addu r0, r1, r2 - R type - Opcode: 0 Function: 2
4: subu r0, r1, r2 - R type - Opcode: 0 Function: 3
5: addi r0, r1, 100 - I type - Opcode: 1
6: addiu r0, r1, 10 - I type - Opcode: 2

3.0.2 Logical Instructions

7: and r0, r1, r2 - R type - Opcode: 3 Function: 0
8: or r0, r1, r2 - R type - Opcode: 4 Function: 0
9: andi r0, r1, 10 - I type - Opcode: 5
10: ori r0, r1, 100 - I type - Opcode: 6
11: sll r0, r1, 10 - R type - Opcode: 7 Function: 0
12: srl r0, r1, 100 - R type - Opcode: 7 Function: 1

3.0.3 Data Transfer Instructions

13: lw r0, 10(r1) - I type - Opcode: 8
14: sw r0, 10(r1) - I type - Opcode: 9

3.0.4 Conditional Branching Instructions

15: beq r0, r1, 10 - I type - Opcode: 10
16: bne r0, r1, 100 - I type - Opcode: 11
17: bgt r0, r1, 10 - I type - Opcode: 12
18: bgte r0, r1, 100 - I type - Opcode: 13
19: ble r0, r1, 10 - I type - Opcode: 14
20: bleq r0, r1, 100 - I type - Opcode: 15

3.0.5 Unconditional Branch Instructions

21: j 100 - J type - Opcode: 16
22: jr r0 - J type - Opcode: 17
23: jal 1000 - J type - Opcode: 18

3.0.6 Comparison Instructions

24: slt r0, r1, r2 - R type - Opcode: 19 Function: 0
25: slti r0, r1, 100 - I type - Opcode: 20

3.0.7 Other Special System Instructions

26: syscall - J type - Opcode: 21
27: display signed integer - System Call Code: 1
28: exit - System Call Code: 2
29: nop - System Call Code: 3
30: display 4 char string - System Call Code: 4
31: display 8 char string - System Call Code: 5
32: display 12 char string - System Call Code: 6
33: display 16 char string - System Call Code: 7
34: display unsigned integer - System Call Code: 8