CS220 Assignment-7
CSE-Bubble Documentation

By Anmol Pabla (190154) and Pranab Pandey (190620)

[PDS1]

We follow a similar register usage protocol to the MIPS-32 Architecture. They have been described in the table below:

Register Number	Name	Usage
0	$zero	Constant 0
1	$at	Reserved for assembler
2-5	$v0-v3	Result registers
6-9	$a0-a3	Argument registers
10-19	$t0-$t9	Temporary registers
20-27	$s0-s7	Saved registers
28	$gp	Global Data Pointer
29	$sp	Stack Pointer
30	$fp	Frame Pointer
31	$ra	Return Address.

Apart from this, we have a PC register that is used to store the address of the next instruction. The bubble sort code to be implemented would require only a few of the registers mentioned above.

[PDS2]

Instruction memory is the portion of memory that holds the program instructions to be executed by the processor. Each instruction in the program is stored in sequential memory locations in the instruction memory. Data memory, on the other hand, is used to store the data that the program operates on, as well as any variables used by the program.

Since we are implementing a bubble sort, it is expected that not more than 200 lines of code in the assembly language would be required, and for a limited array size, 100-word storage spaces should suffice. Thus, the **instruction memory has a size of 800 bytes** (200x4), and the **data memory has a size of 400 bytes** (100x4). Thus, the VEDA_inst contains 200 rows and VEDA_data contains 100 rows.

[PDS3]

We follow the MIPS-32 architecture for the design layout of the instructions. The instructions given can be divided into three categories as given below:

- R-type - ***add, sub, addu, subu, and, or, sll, srl, slt***
- I-type - ***addi, addiu, andi, ori, slti, lw, sw, beq, bne, bgt, bgte, ble, bleq***
- J-type - ***j, jr, jal***

The encoding styles of the three instruction formats are given below:

R-type-

op(6 bits)	rs(5 bits)	rt(5 bits)	rd(5 bits)	shamt(5 bits)	funct(6 bits)

- Registers rs, rt are operand registers, rd is the destination register, op and funct help identify the operation and shamt is the shift amount.
- op and funct codes for each instruction are given below.

Instruction	op code	funct
add	0	32
sub	0	33
addu	0	34
subu	0	35

and	0	36
or	0	37
sll	0	38
srl	0	39
slt	0	40

I-type-

op(6 bits)	rs(5 bits)	rd(5 bits)	Immediate value(16 bits)

- Registers rs is the operand register, rd is the destination register, op helps identify the operation and immediate value is the constant value used in operation.
- op codes for each instruction are given below.

Instruction	op code
addi	8
addiu	9
andi	10
ori	11
slti	12
lw	13
sw	14
beq	15
bne	16
bgt	17
bgte	18
ble	19

bleq	20

J-type-

op(6 bits)	address(26 bits)

- op helps identify the instruction, address contains the address to be jumped to and in case of jr the address of the register that contains the address to be jumped to.
- op codes for each instruction are given below.

Instruction	op code
j	1
jr	2
jal	3

[PDS4] - [PDS10]

PDS4 to PDS10 were implemented as code:

- PDS4 to PDS8 were implemented in verilog along with their test benches.
- the MIPS code for bubble sort was implemented in PDS9, which was then simplified and converted into machine code according to the above instruction set architecture and loaded into VEDA_inst.