**CS210- Lab 12:**

**Study of multi- Cycle Processor/Pipelined MIPS processor**

**Task 1: Study the given pipeline implementation of the processor, and identity error in the design (if any) . Convert the following C code to MIPS assembly, test the code in MIPS pipelined version(Given) .**

**for(i = 0; i < 100; i++)**

**A[i] = i;**  (**30 points)**

Ans :-

Mips Code –

.data

array : .word 0

.text

addiu $s0,$s0,100

addiu $s7,$s7,1

addiu $s5,$s5,0

loop:

beq $s7,$s0,exit

sw $s7,0($s1)

addiu $s1,$s1,4

addiu $s7,$s7,1

beq $0,$0,loop

exit:

Machine Hexadecimal instruction for above mips code are:

.

v2.0 raw

26100064

26f70001

26b50000

12f00004

ae370000

26310004

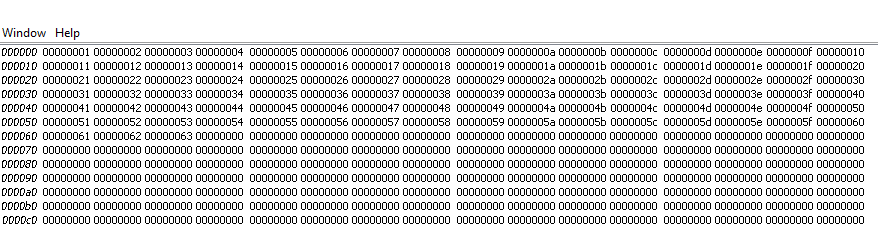
26f70001

1000fffb

In Memory we can see it is storing the values from 1 to 100

Testing code in mips pipelined version :

Below is memory in logisim.



Remarks : In mars memory start from 0x10010000

And in logisim we have memory element from 0000

**Task 2:** ***Write ASCII values of your name full name to the memory using appropriate instruction, and write program to find the sum of those values using the given pipeline MIPS design. Computer the number of cycles required and compare with number of cycles that you got in previous(Multi-cycle implementation).* Enter the relevant information in the r*ecord* for demonstration*.***

**(30 points)**

Character : A S I F HUSSAIN

Ascii Values : 65 78 83 72 85 32 78 65 85 87 65 76 65

Hexadec val: 41 4E 53 48 55 20 4E 41 55 57 41 4C 41

addiu $t2 $t2 65

addiu $t3 $0 65

sw $t3 4($0)

addiu $t2 $t2 78

addiu $t3 $0 78

sw $t3 8($0)

addiu $t2 $t2 83

addiu $t3 $0 83

sw $t3 12($0)

addiu $t2 $t2 72

addiu $t3 $0 72

sw $t3 16($0)

addiu $t2 $t2 85

addiu $t3 $0 85

sw $t3 20($0)

addiu $t2 $t2 32

addiu $t3 $0 32

sw $t3 24($0)

addiu $t2 $t2 78

addiu $t3 $0 78

sw $t3 28($0)

addiu $t2 $t2 65

addiu $t3 $0 65

sw $t3 32($0)

addiu $t2 $t2 85

addiu $t3 $0 85

sw $t3 36($0)

addiu $t2 $t2 87

addiu $t3 $0 87

sw $t3 40($0)

addiu $t2 $t2 65

addiu $t3 $0 65

sw $t3 44($0)

addiu $t2 $t2 76

addiu $t3 $0 76

sw $t3 48($0)

addiu $t2 $t2 65

addiu $t3 $0 65

sw $t3 44($0)

sw $t2 36($0)

.dat file

v2.0 raw

254a0041

240b0041

ac0b0004

254a004e

240b004e

ac0b0008

254a0053

240b0053

ac0b000c

254a0048

240b0048

ac0b0010

254a0055

240b0055

ac0b0014

254a0020

240b0020

ac0b0018

254a004e

240b004e

ac0b001c

254a0041

240b0041

ac0b0020

254a0055

240b0055

ac0b0024

254a0057

240b0057

ac0b0028

254a0041

240b0041

ac0b002c

254a004c

240b004c

ac0b0030

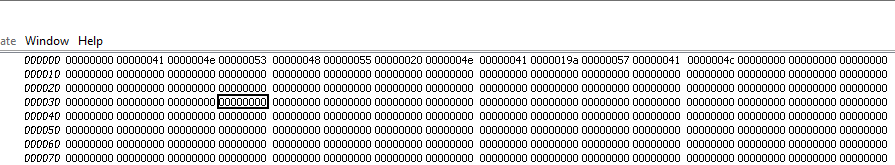
254a0041

240b0041

ac0b002c

ac0a0024

**Memory of logisim:**

****

**Task 3: Add one new instructions to the given architecture and test using new test program.**

**(40 points)**

**Example average of two numbers**

**Mips code using real instructions:**

**Negative of a Number :**

Opcode 0x1C: This opcode was reserved for the LSA (Load String and Add) instruction in the MIPS32/64 Release 2 ISA, but it was never implemented.

In the MIPS64 Release 6 ISA, this opcode is used for the WSBH (Word Swap Bytes within Halfwords) instruction.

We create new instruction ‘neg’ which will use this opcode;

Neg will be an i-type instruction and neg rt, rs will store the negative of the number in rs to rt;

Immediate can be completely ignored.

Neg $t1, $t0, 0=011100|01000|01001|0000000000000000

**Submission (modified file and test files) :**

[https://u.pcloud.com/#page=puplink&code=YHwkZXumzadobsxQuqiGEkouWoft2M4LX](https://u.pcloud.com/" \l "page=puplink&code=YHwkZXumzadobsxQuqiGEkouWoft2M4LX)

Due: 12.30 PM, 28th March 2023.