**Design Sheet**

# **Team Members**

Team member: Gabe Smith

Team member: Keyona Robertson

# **Design Overview**

Explain the set of instructions, clocking methodology (falling or rising edge triggered), registers file, and other specifications

Instructions:

R type (5):

add

sub

and

or

sll

srl

I type (5):

addi

lb

sb

lw

sw

beq

bne

J type (3):

J

JAL

Bonus instructions:

Clocking methodology: This processor will run on a 20MHz which will be rising edge triggered.

## **Instruction format**

Example:

R-Format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Opcode | rs | rt | rd | Func |
| 4 bits | 3 bits | 3 bits | 3 bits | 3 bits |

I–Format:

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Rs | Rt | Imm |
| 4 bits | 3 bits | 3 bits | 6 bits |

J- Format:

|  |  |
| --- | --- |
| Opcode | address |
| 4 bits | 12 bits |

## **Instructions**

Example:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Mnemonic | Operation | Opcode | Func | Format |
| Add | add | add $s1, $s2, $s3; $s1 = $s2 + $s3 | 0000 (0x0) | 000 | R |
| Sub | sub | sub $s1, $s2, $s3; $s1 = $s2 - $s3 | 0001 (0x1) | 000 | R |
| Bitwise And | and | and $s1, $s2, $s3; $s1 = $s2 & $s3 | 0010 (0x2) | 000 | R |
| Bitwise Or | or | or $s1, $s2, $s3; $s1 = $s2 | $s3 | 0011 (0x3) | 000 | R |
| Shift left logical | sll | sll $s1, $s2, 2; $s1 = $s2 << 2 | 0100 (0x4) | 000 | R |
| Shift right logical | srl | srl $s1, $s2, 2; $s1 = $s2 >> 2 | 0101 (0x5) | 000 | R |
| Add immediate | addi | addi $s1, $s2, 100; $s1 = $s2 + 100 | 0110 (0x6) | 000 | I |
| Load byte | lb | lb $t1, offset($s1); $t1 = MEM[$s1 + offset] | 0111 (0x7) | 000 | I |
| Store byte | sb | sb $t1, offset($s1); MEM[$s1 + offset] = (0xff + $t1) | 1000 (0x8) | 000 | I |
| Load word | lw | lw $t1, offset($s1); $t1 = MEM[$s1 + offset] | 1001 (0x9) | 000 | I |
| Store word | sw | sw $t1, offset($s1); MEM[$s1 + offset] = $t1 | 1010 (0xA) | 000 | I |
| Branch on equal | beq | beq $s1, $t1, offset; if $s1 == $t1 advance\_pc (offset << 2) | 1011 (0xB) | 000 | I |
| Branch on not equal | bne | bne $s1, $t1, offset; if $s1 != $t1 advance\_pc (offset << 2) | 1100 (0xC) | 000 | I |
| Jump | j | j target; PC = nPC | 1101 (0xD) | 000 | J |
| Jump and link | jal | jal target; $31 = PC + 4 | 1110 (0xE) | 000 | J |

## **Assembly language and machine code for the test program (Pseudocode)**

Pseudocode for the test program (refer to the project handout):

while ($a1>0) do

{

……

}

|  |  |
| --- | --- |
| Assembly Language | Machine Code |
| add $3, $1, $2 DONE | 0x0650: 0000 0110 0101 0000 |
| sub $4, $1, $2 DONE | 0x1850: 0001 1000 0101 0000 |
| and $5, $4, $1 | 0x2B08: 0010 1011 0000 1000 |
| or $1, $5, $3 | 0x3358: 0011 0011 0101 1000 |
| sll $4, $3, 2 | 0x48B0: 0100 1000 1101 0000 |
| srl $2, $3, $4 | 0x54E0: 0101 0100 1110 0000 |
| addi $2, $3, 4 DONE | 0x64C4: 0110 0100 1100 0100 |
| lb $1, 0($2) | 0x7440: 0111 0100 0100 0000 |
| sb $2, 0($3) | 0x8680: 1000 0110 1000 0000 |
| lw $3, 0($4) | 0x9700: 1001 0111 0000 0000 |
| sw $2, 50($5) | 0xAAB2: 1010 1010 1011 0010 |
| beq $5, $5, 12 | 0xBB4C: 1011 1011 0100 1100 |
| bne $4, $2, 13 | 0xC88B: 1100 1000 1000 1101 |
| j | 0xD: |
| jal n\*(PC+4) | 0xE: |

\*\* You can explain your assembly language in brief if needed.