SIMPLE RISC-LIKE ARCHITECTURE

SAMPLE MICROPROCESSOR

SPECIFICATIONS OF THE CPU:

CLOCK TIME PERIOD = 6 ns

HARDWIRED CONTROL UNIT

NO PIPELINING IMPLEMENTED (UPCOMING)

GENERAL PURPOSE REGISTERS

16 GENERAL PURPOSE 16 BIT REGISTERS

THESE 16 REGISTERS INCLUDE ACCUMULATOR AND DATA REGISTER WHICH FEEDS THE ALU.

SPECIAL PURPOSE REGISTERS

PROGRAM COUNTER

STACK POINTER

-----------------------------------------------------------------------------------------------

MEMORY CHIPS:

DATA MEMORY (35536 X 16) = 16 BIT ADDRESS (2^16 MEMORY LOCATIONS 16 BIT EACH)

INSTRUCTION MEMORY (35536 X 32) = 16 BIT ADDRESS (2^16 MEMORY LOCATIONS 32 BIT EACH)

-----------------------------------------------------------------------------------------------

ADDRESSING MODES

1. REGISTER ADDRESSING (ALL OPERANDS ARE REGSITERS)
2. IMMEDIATE ADDRESSING (ONLY 1 IMMEDIATE ALLOWED)
3. BASE ADDRESSING
4. DIRECT ADDRESSING
5. PC RELATIVE ADDRESSING
6. STACK ADDRESSING

ISA AND OP CODE ALLOTMENT

**ALU INSTRUCTIONS**

**ADDRESSING MODE 1: BASIC ALU OPERATIONS BETWEEN 2 REGISTERS & STORE IN A REG (CLASS 1)**

* **SOURCE OPERAND: ANY 2 REGISTERS**
* **DESTINATION OPERAND: ANY 1 REGISTER**
* **OPERATOR: +, -, \*, /, %, &&, ||, ^^, ~~**
* **CLOCK CYCLES NEEDED: 6**
* **AT THE END OF THE SECOND CLOCK CYCLE (T [1]), IR GETS THE INSTRUCTION TO BE EXECUTED.**
* **CYCLE 3: OUTPUT ENABLE OF SOURCE 1 AND LOAD ENABLE OF ACC**
* **CYCLE 4: OUTPUT ENABLE OF SOURCE 2 AND LOAD ENABLE OF DR. ALSO OUTPUT ENABLE OF ACC AND DR TO FETCH OPERANDS TO ALU**

**ALU COMPUTES IMMEDIATELY...........**

* **CYCLE 5: LOAD ENABLE OF ACC**
* **CYCLE 6: OUTPUT ENABLE OF ACC AND LOAD ENABLE OF DESTINATION**

**LIST OF OPERATIONS:**

000000 = ADD WITH CARRY (CARRY STORED IN CARRY FLAG)

000001 = SUB WITH CARRY (AUXILLIARY CARRY DENOTED IN AUX CARRY FLAG, CARRY DENOTED IN CARRY FLAG AND OVERFLOW DENOTED IN OVERFLOW FLAG)

000010 = MUL (MUL MUST STORE HIGHER 32 BITS IN ACCUMULATOR, AND LOWER 32 BITS IN DR)

000011 = DIV (DIV MUST STORE QUOTIENT IN ACCUMULATOR, AND REMAINDER IN DR)

000100 = BITWISE AND

000101 = BITWISE XOR

000110 = BITWISE OR

000111 = BITWISE NOT

**INSTRUCTION DETAILS:**

* IR [3:0] --> SOURCE REG 1
* IR [7:4] --> SOURCE REG 2
* IR [11:8] --> DEST REG

NOTE: FOR THE REGISTER BANK, LOAD ENABLE AND OUTPUT ENABLE WILL GET OUTPUT OF MUX AS MANY SIGNALS WILL CONDITIONALLY DRIVE THE PORTS.

**ADDRESSING MODE 2: BASIC ALU OPS BETWEEN A REG AND AN IMMEDIATE & STORE IN A REG (CLASS 2)**

* **SOURCE OPERAND: ANY 1 REGISTER AND 1 IMMEDIATE**
* **DESTINATION OPERAND: ANY 1 REGISTER**
* **OPERATOR: +, -, \*, /, %, &&, ||, ^^, ~~**
* **CLOCK CYCLES NEEDED: 6**

**LIST OF OPERATIONS:**

100000 = ADD WITH CARRY (CARRY STORED IN CARRY FLAG)

100001 = SUB WITH CARRY (AUXILLIARY CARRY DENOTED IN AUX CARRY FLAG, CARRY DENOTED IN CARRY FLAG AND OVERFLOW DENOTED IN OVERFLOW FLAG)

100010 = MUL (MUL MUST STORE HIGHER 32 BITS IN ACCUMULATOR, AND LOWER 32 BITS IN DR)

100011 = DIV (DIV MUST STORE QUOTIENT IN ACCUMULATOR, AND REMAINDER IN DR)

100100 = BITWISE AND

100101 = BITWISE XOR

100110 = BITWISE OR

100111 = BITWISE NOT

**INSTRUCTION DETAILS:**

* IR [3:0] --> SOURCE REG 1
* IR [7:4] --> DEST REG
* IR [23:8] --> IMMEDIATE VALUE

**CLASS 3: LS RS ARS OF A REG AND INCLUDE NOP/ THINK ABOUT LEFT SHIFTING BIT OF ANOTHER REGISTER INTO GIVEN REGSITER**

**OPCODE DETAILS:**

OPCODE [5] = 0 (REG ADDRESSING)

OPCODE [4:3] =01

**OPCODES LIST:**

001000 = CIRCULAR LEFT SHIFT

001001 = CIRCULAR RIGHT SHIFT

001010 = ARITHMETIC RIGHT SHIFT

001011 = NOP

**INSTRUCTION DETAILS:**

* IR [3:0] --> SOURCE REG 1

**REGISTER TRANSFER AND IMMEDIATE DATA LOADING**

**ADDRESSING MODE 2: IMMEDIATE DATA LOAD INTO A REG (CLASS 4)**

**OPCODE DETAILS:**

OPCODE [5] = 1 (IMM ADDRESSING)

OPCODE [4:3] =10

OPCODE [2:0] =000

**OPCODES LIST:**

110000

**INSTRUCTION DETAILS:**

* IR [3:0] --> DEST REG
* IR [8:23] --> IMMEDIATE VALUE

**ADDRESSING MODE 1: TRANSFER BETWEEN 2 REGISTERS (CLASS 5)**

**OPCODE DETAILS:**

OPCODE [5] = 0 (REG ADDRESSING)

OPCODE [4:3] =10

OPCODE [2:0] =000

**LIST OF OPERATIONS:**

010000

**INSTRUCTION DETAILS:**

* IR [3:0] --> SRC REG
* IR [7:4] --> DEST REG

**MEMORY LOAD & STORE**

10001 = LOAD FROM MEMORY LOCATION IN R2 TO R1

10010 = STORE INTO MEMORY LOCATION IN R2 FROM R1

**ADDRESSING MODE 4: STORE SOME DATA FROM ANY REG TO ANY MEMORY LOCATION (CLASS 6)**

**OPCODE DETAILS:**

OPCODE [5] = 0 (REG ADDRESSING)

OPCODE [4:3] =01

OPCODE [2:0] =100

**LIST OF OPERATIONS:**

001100

**INSTRUCTION DETAILS:**

* IR [3:0] --> SRC REG
* IR [8:23] --> ADDRESS IN DATA MEMORY

**ADDRESSING MODE 4: LOAD SOME DATA FROM ANY MEMORY LOCATION TO ANY REG (CLASS 7)**

**OPCODE DETAILS:**

OPCODE [5] = 0 (REG ADDRESSING)

OPCODE [4:3] =01

OPCODE [2:0] =101

**LIST OF OPERATIONS:**

001101

**INSTRUCTION DETAILS:**

* IR [3:0] --> SRC REG
* IR [8:23] --> ADDRESS IN DATA MEMORY

**BRANCH INSTRUCTIONS**

**ADDRESSING MODE 5: UNCONDITIONAL BRANCH (CLASS 8)**

**OPCODE DETAILS:**

OPCODE [5] = 0 (REG ADDRESSING)

OPCODE [4:3] =01

OPCODE [2:0] =110

**OPCODES LIST:**

001110

**INSTRUCTION DETAILS:**

* IR [23:8] --> OFFSET

**ADDRESSING MODE 5: JUMP/ BRANCH IF TWO REGISTERS ARE EQUAL (CLASS 9)**

**OPCODE DETAILS:**

OPCODE [5] = 0 (REG ADDRESSING)

OPCODE [4:3] =01

OPCODE [2:0] =110

**OPCODES LIST:**

001110

**INSTRUCTION DETAILS:**

* IR [3:0] --> SRC REG 1
* IR [7:4] --> SRC REG 2
* IR [15:0] --> OFFSET

**ADDRESSING MODE 5: JUMP/ BRANCH IF REGISTER A > 0 (CLASS 10)**

**OPCODE DETAILS:**

OPCODE [5] = 0 (REG ADDRESSING)

OPCODE [4:3] =01

OPCODE [2:0] =110

**OPCODES LIST:**

001110

**INSTRUCTION DETAILS:**

* IR [3:0] --> SRC REG 1
* IR [7:4] --> SRC REG 2
* IR [15:0] --> OFFSET

**ADDRESSING MODE 5: JUMP/ BRANCH IF REGISTER A < 0 (CLASS 11)**

INTERFACE WITH THE TIMER

TIMER IS NEGATIVE EDGE TRIGGERED

T0:

* INSTRUCTION MEMORY ADDRESS REGISTER LOAD ENABLE
* ENABLE PC INCREMENTER (OUTPUT OF PC CONNECTED TO ADDER)
* PC LOAD ENABLE

***THIS ARRANGEMENT WILL NOT WORK IF DELAYS ARE INTRODUCED***

**NOTE: IMAR IS GETTING THE VALUE NEEDED FOR THAT EXECUTION CYCLE AT THE MIDPOINT OF T0.**

T1:

* INSTRUCTION MEMORY ACCEESS (READ ENABLE) (MEMORY IS NEGATIVE EDGE TRIGGERED)
* LOAD ENABLE OF IR (POSITIVE EDGE TRIGGERED)

NOTE:

**NOTE: IR IS GETTING THE INSTRUCTION NEEDED FOR THAT EXECUTION CYCLE AT THE MIDPOINT OF T1.**

T2:

PROGRAM STATUS WORD:

16 BITS

PSW [0] = Carryout of ALU

PSW [1] = Auxiliary Carryout of ALU

PSW [2] = Overflow of ALU

PSW [3] = A = B

PSW [4] = Overflow of ALU

PSW [5] = Overflow of ALU

**USED OPCODES**

000000 = ADD

000001 = SUB

000010 = MUL

000011 = DIV

000100 = AND

000101 = XOR

000110 = OR

000111 = NOT

100000 = ADD

100001 = SUB

100010 = MUL

100011 = DIV

100100 = AND

100101 = XOR

100110 = OR

100111 = NOT

**TO DO:**

1. **STUDY 2’S COMPLEMENT ARITHMETIC IN DETAILS (OVERFLOW, CARRY, BORROW AND AUXILLIARY CARRY)**
2. **INCLUDE A MUX AT THE DATA INPUT OF THE PC WHICH IS DRIVEN BY TIME STATE**

* **STORE NORMALLY INCREMENTED ADDRESS INTO PC)**
* **IF BRANCH COMES, UPDATE THE PC BY TAKING OUTPUT FROM ALU**

1. **IMPLEMENTATION OF NOP LEFT**
2. **PSW IMPLEMENTATION LEFT**
3. **TEST CLASS 6,7**
4. **IMPLEMENT CLASS 8 ONWARDS**