24 Bit Instruction based Processor

Rivier University

CS556AH2 Computer Architecture

Fall 2016

Submitted to: Robert Marceau, Ph.D.

Submitted by:

Chaitrali Jayantilal Doshi

Table of Contents

1 Purpose

The purpose of the project is to design a processor which is also capable of performing dot product operation on 2 arrays.

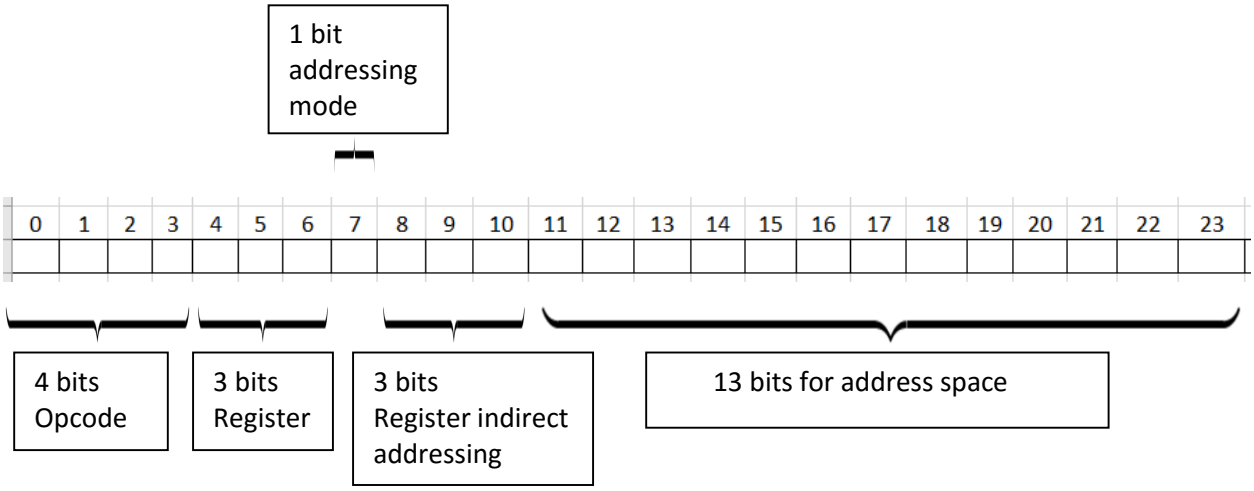
2 Specification

- Word Size: 24 Bit, Word addressable
- Endianness: Big Endian
- Number and Type of registers: 8 Register and General Purpose Register
- Address Space: 13 bits
- Addressing modes implemented: Direct and Indirect addressing (@ symbol for indirect addressing)
- Number of opcodes: 16 opcodes

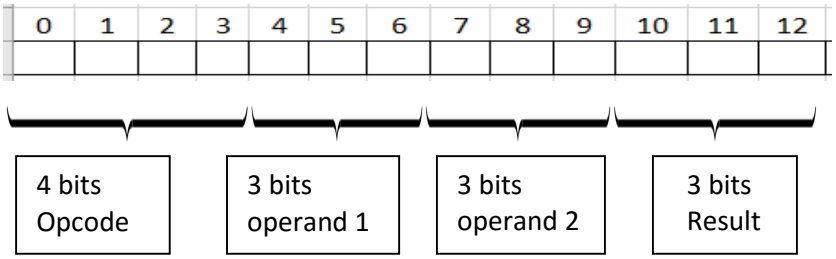
2.1 List of Opcodes

1. LOAD
2. STORE
3. INC
4. DEC
5. CLR
6. COMPARE
7. SKIPEQ
8. SKIPGT
9. SKIPLT
10. JUMP
11. HALT
12. ADD
13. SUBT
14. MULT
15. DIV
16. MOD

2.2 Instruction format for opcode 1 to 11



2.3 Instruction format for opcode 12 to 16



2.3.1 Opcode Description

1. LOAD: This instruction loads the data from given operand into the register.

Example:
LOAD R1, counter

2. STORE: This instruction stores the data from given operand into the register. If the value of given register is needed further in the program, then store instruction can be used.

Example:
STORE R1, counter

3. INC: This instruction is used to increment the register counter. This can be used in case loop condition or while accessing array elements etc.

Example:
INC R1

4. DEC: This instruction is used to decrement the register counter.

Example:
DEC R1

5. CLR: As a name indicates CLR instruction is used to clear the value from register.

Example:
CLR R1

6. COMPARE: This instruction is used for comparing value of two registers or register and operand and gives the result as below:

```
00: if equal

01: if less than

10: if greater than
```

Example:
COMPARE R0, SIZE, R7

7. SKIPEQ: This instruction is used for skipping the next instruction if the given register's value is equal to value 00.

Example:
SKIPEQ R1

8. SKIPLT: This instruction is used for skipping the next instruction if the given register's value is equal to value 01.

Example:
SKIPLT R1

9. SKIPGT: This instruction is used for skipping the next instruction if the given register's value is equal to value 10.

Example:
SKIPGT R1

10. JUMP: This instruction is used for reaching the code directly by providing the address.

Example: Here loop is the label.
JUMP LOOP

11. HALT: This instruction is used for stopping the program execution.

Example:
HALT

12. ADD: This instruction is used for performing summation of two operands and stores results in the third operand.

Example:
ADD R1, R2, R3

13. SUBT: This instruction is used for performing subtraction of two operands and stores results in the third operand.

Example:
SUB R1, R2, R3

14. MULT: This instruction is used for performing multiplication of two operands and stores results in the third operand.

Example:
MULT R1, R2, R3

15. DIV: This instruction is used for performing division of two operands and stores results in the third operand.

Example:
DIV R1, R2, R3

16. MOD: This instruction is used for performing modulo of two operands and stores results in the third operand.

Example:
MOD R1, R2, R3

3 Dot Product Example

```
START: LOAD R0, CNT    // loads value of CNT to R0
       LOAD R1, ADDR1  // loads value of ADDR1 to R1
       LOAD R2, ADDR2  // loads value of ADDR2 to R2
         CLEAR R3        // clear R3 for storing final result
         CLEAR R6          // clear R6 for storing product result
LOOP:  LOAD R4, @R1    // loads element of the array A
       LOAD R5, @R2    // loads element of the array B
         MULT R4, R5, R6  // multiply R4,R5 and stores into R6
         ADD R6, R3, R3 // add R6,R3 and stores into R3
         INC R0 //incrementing R0
         INC R1
         INC R2
         COMPARE R0, SIZE, R7  // 00:equal, 01:less than, 10: greater than
         SKIPEQ R7             // Skips next instruction if 00
         JUMP LOOP      //jumping at loop label
STOP   HALT

ADDR1: A
ADDR2: B
SIZE: 5
CNT: 0
A: 1
   3
   5
   7
   9
B: 2
   4
   6
   8
   10
```

Notes:

'@' symbol is used for indirect addressing

'//' symbol is used for comments

Label A, B are the pointing to the first element of the array A and B.

ADDR1 and ADDR2 are pointers to the array locations

4 Conclusion

We were able to design a processor which is capable of performing dot product operation on 2 arrays.

5 References

Textbook: Linda Null and Julia Lobur, The Essentials of Computer Organization and Architecture, 4th Ed.; Jones and Bartlett; 2014,