Bibodi Jay N

05/13/2016

CSC 35

Section 1

ManGo Core Processor

# Design

16-bit Mango Core processor uses little endian approach to store bytes in memory. It has 16 General Purpose Register which will require 4 bits for each register code. This processor uses 2 operands to perform all operations. Therefore, in total 8 bits or 1 byte is used to store or load 2 operands in Mango Core Processor. It is designed to be simple and efficient in terms of hardware and hence it follows Reduced Instruction Set Computer abbreviated as RISC approach.

More operands give greater functionality but require higher bits to store in memory and so is the reason 16-bit Mango Core Processor uses average number of operands that is 2 operands through which we can achieve all useful functionality and store appropriate bits in memory.

One of the main reason to use RISC approach is that Instruction Set can be simpler can be used easily on other processor. Majority of operations are performed using register which minimizes use of memory and instruction can be carried by at higher speed.

16 General Purpose register are as follow

|  |  |  |  |
| --- | --- | --- | --- |
| **Register Name** | **Register Code** | **Register Name** | **Register Code** |
| R0 | 0000 | R8 | 1000 |
| R1 | 0001 | R9 | 1001 |
| R2 | 0010 | R10 | 1010 |
| R3 | 0011 | R11 | 1011 |
| R4 | 0100 | R12 | 1100 |
| R5 | 0101 | R13 | 1101 |
| R6 | 0110 | R14 | 1110 |
| R7 | 0111 | R15 | 1111 |

# Instructions

## MOV data from Immediate to register

We store an 8-bit immediate into 2nd Operand. Note that this consumes 16 different opcodes. Suppose MOV immediate to R0, R1 and so on. So, it consumes 0000 0000, 0000 0001, 0000 0010, and 0000 0011 etc. The format shown below one highlighted with yellow describes operation code (in this case its mov operation) and one highlighted with green shows R0 register where immediate value is moved. The 8 bits from LSB contains immediate value.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | R1 | | | | Immediate Value | | | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## MOV data from register to register

Processor move value from one register to another. Area highlighted with blue contains register code for R0 (0000) and area highlighted with green contains register code for another register in this case its register R1 (0001). Area highlighted with yellow color describe move operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | R0 | | | | R1 | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## Load Memory to Register using Register Indirect

Processor load address from memory into register. Area highlighted with blue contains register code for R0 (0000) which holds value that is stored as effective address and area highlighted with green contains register code for another register which have value from memory stored in register, in this case its register R2 (0010). Area highlighted with yellow color describe load operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | R0 | | | | R2 | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## Store Register value into Memory using Register Indirect

Processor stores value from one register to another which contains address of the memory. Area highlighted with blue contains register code for R3 (0011) which holds the value of address in memory and area highlighted with green contains register code for another register which have value to be stored at memory location that is described in register R3. The register which contains value is R4(0100). Area highlighted with yellow color describe store operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | R3 | | | | R4 | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## Compare Data in two register

Compare the values in two register highlighted with blue and green and based on that flags are set which are used for conditional jumping. Area highlighted with yellow color describe compare operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | R3 | | | | R4 | | | |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## Jump

Jump relative to program counter based on offset. The offset is signed 8-bit integer contains the target address to move. Area highlighted with yellow color describe jump operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | offset | | | | | | | |
| 0 | 0 | 0 | 0 | 1 | X | y | z |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

Where xyz is,

000 = jump equals

001 = jump not equals

010 = jump greater than

011 = jump greater than or equal to

100 = jump less than

101 = jump less than or equal to

110 = default jump when you just want to jump out of the loop without comparing data or just moving to some other memory location to execute that particular piece of logic.

## Add values using two register

Add the values from two register highlighted with blue (Register R8) and green (Register R9) and result is stored R9. Area highlighted with yellow color describe addition operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | R8 | | | | R9 | | | |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## Subtract values using two register

Subtract the values from two register highlighted with blue (Register R8) and green (Register R9) and result is stored R9. Area highlighted with yellow color describe subtract operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | R8 | | | | R9 | | | |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## Multiply values using two register

Multiply the values from two register highlighted with blue (Register R8) and green (Register R9) and result is stored R9. Area highlighted with yellow color describe multiplication operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | R8 | | | | R9 | | | |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## Divide values using two register

Divide the values from two register highlighted with blue (Register R8) and green (Register R9) and quotient is stored R9. Area highlighted with yellow color describe division operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | R8 | | | | R9 | | | |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## Mod operation using two register

Find reminder when value from one register highlighted with blue area (Register R8) is divided by value from second register highlighted with green area (Register R9) and result is stored R9. Area highlighted with yellow color describe modulus operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | R8 | | | | R9 | | | |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |

## Software Interrupt

Interrupt the system using vector number. Area highlighted with yellow color describes interrupt operation to be performed by processor.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | | | | | | | | Vector Number | | | | | | | |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |
| 15 |  |  |  |  |  |  | 8 | 7 |  |  |  |  |  |  | 0 |