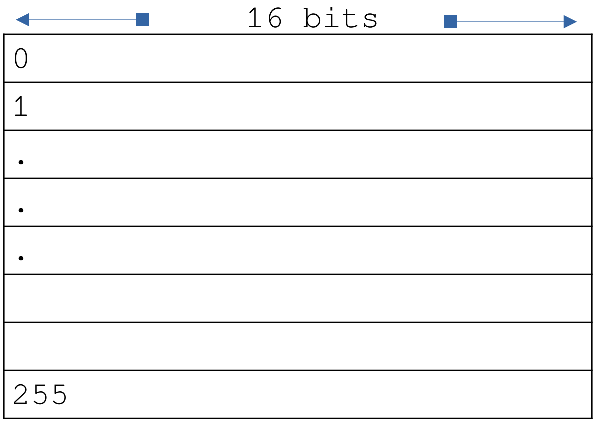
Henok Belayneh UGR/3086/14

A 4 BIT CPU DESIGN USING LOGISM

The Design Specifications of the CPU

- 4 bit CPU
- RISC Micro-controller Architecture
- 8 general purpose Registers
  - **R_0** through **R_7** each with a 4 bit size
- 3 bus architecture A, B, and C(in which the ALU, registers,the program counters,Instruction Register and Decoder Blocks would be around.)
- **4** Instructions:
  - MovImd, #number, **R_D**
  - ADD, **R_A, R_B, R_D :- R_A + R_B => R_D**
  - AND, **R_A, R_B, R_D :- R_A, & R_B => R_D**
  - OR, **R_A, R_B, R_D :- R_A, || R_B => R_D**
- Program Memory 256 X 16
-



  - One Instruction Requires one word(Every Location Stores a 16 bit number, in which the instruction requires one word of 16 bit which has to be stored in every address.)
  - Word Addressable Memory(so it is 2 bytes/16 bits addressable)

Machine coding Instructions

The 15th and 14th bits represent the opcode

- **0 0 for ADD**
- **0 1 for AND**
- **1 0 for OR ,and**
- **1 1 for MovImd**

for example the following table represents the machine code for the *MovImd* instruction

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	X	X	X	X	X	X	X							

op code 2 bits

Numbers 4 bits

Rd = regiater Destination 3 bits

The rest of the bits from *bit 7 to 13* are null with inputs of 0000, which we will not be using for the time being.
Bits 0 ,1, and 2 are reserved to represent the Registers
Each register holds a number of 4 bits width from **0000** through **1111**

3 bits means 2^3 = 8 binary combinations

RD	Binary representation
R0	000
R1	001
R2	010
R3	011
R4	100
R5	101
R6	110
R7	111

We will use a hexadecimal code for the numbers

BINARY	1111	1110	1101	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	0000
HEXADECIMAL	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Example1:- **MovImd #4, R4**

1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0

MovImd 4 R4

But we have to divide the above table in the example in to groups of 4 bits and represent each one with a hexadecimal code format

1	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0

c 0 2 4

c024

So ***c024*** represents moving the number 4 to the 4th Register, R4.

Data manipulation Instructions

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0														

op code for data manipulation instruction

OPeration
00 = ADD
01 = AND
10 = OR
11 = XOR

RA RB RD

The data manipulation instructions are the instructions that will be represented with the 16 bits, after which they are groped in to groups of four and changed to a hexadecimal code format from left right.
The three bits from namely 9, 10, and 11 are null bits that are extra and we will not use.
The 15 and 14 bits being 0 0 indicates a data manipulation instruction and the 13 and 12 bits specify what type of data manipulation instruction the ALU will be performing.

Example2:- ADD , R4, R2, R7 or R4 + R2 => R7

1 7

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	1

0 1

0117

so ***0117 in hexadecimal*** represents adding R4 and R2 and Storing the result in R7

Example 3:- XOR R4, R2, R7

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	1	0	0	0	1	0	1	1	0

3116

Store in the program Memory

every instruction is one clock cycle
so 4 clock cycles will be enough to finish the execution of the whole program.

ALP code		MLP code
Mov #4, R4	0	c024
Mov #5, R2	1	c02A
ADD , R4, R2, R7	2	0117
XOR R4, R2, R7	3	3116
.	.	.
.	.	.
.	.	.
	255	