Layla Johnson

Dr. Omar El Khatib

Computer Organization

12-11-25

Opcode & Function Tables

Instruction	Opcode	Funct
ADD	0x00	0x20
SUB	0x00	0x22
ADDI	0x08	n/a
BEQ	0x04	n/a
LW	0x23	n/a
SW	0x2B	n/a

Instruction	**Type**	**Opcode**	**Funct**	**rs**	**rt**	**rd**	**imm**
ADD	R	0x00	0x20	used	used	used	n/a
SUB	R	0x00	0x22	used	used	used	n/a
ADDI	I	0x08	n/a	used	used	n/a	used
BEQ	I	0x04	n/a	used	used	n/a	used
LW	I	0x23	n/a	used	used	n/a	used
SW	I	0x2B	n/a	used	used	n/a	used

Encoding Examples

1. Binary = 001000 00000 01000 0000000000000101 Hex = 0x20080005
2. Binary = 001000 00000 01001 0000000000000011 Hex = 0x20090003
3. Binary = 000000 01000 01001 01010 00000 100000 Hex = 0x01095020

Project Report

This project implements a mini MIPS-like CPU in C++. The simulator includes an instruction set that is encoded into binary and hexadecimal, loaded into a simulated memory, and executed through instruction fetch, decode, execute, memory access, and write-back.

Test Program

The test program (test.hex) initializes registers R8 and R9 using immediate values, adds their contents, and stores the result to memory address 0. The hexadecimal instructions are:

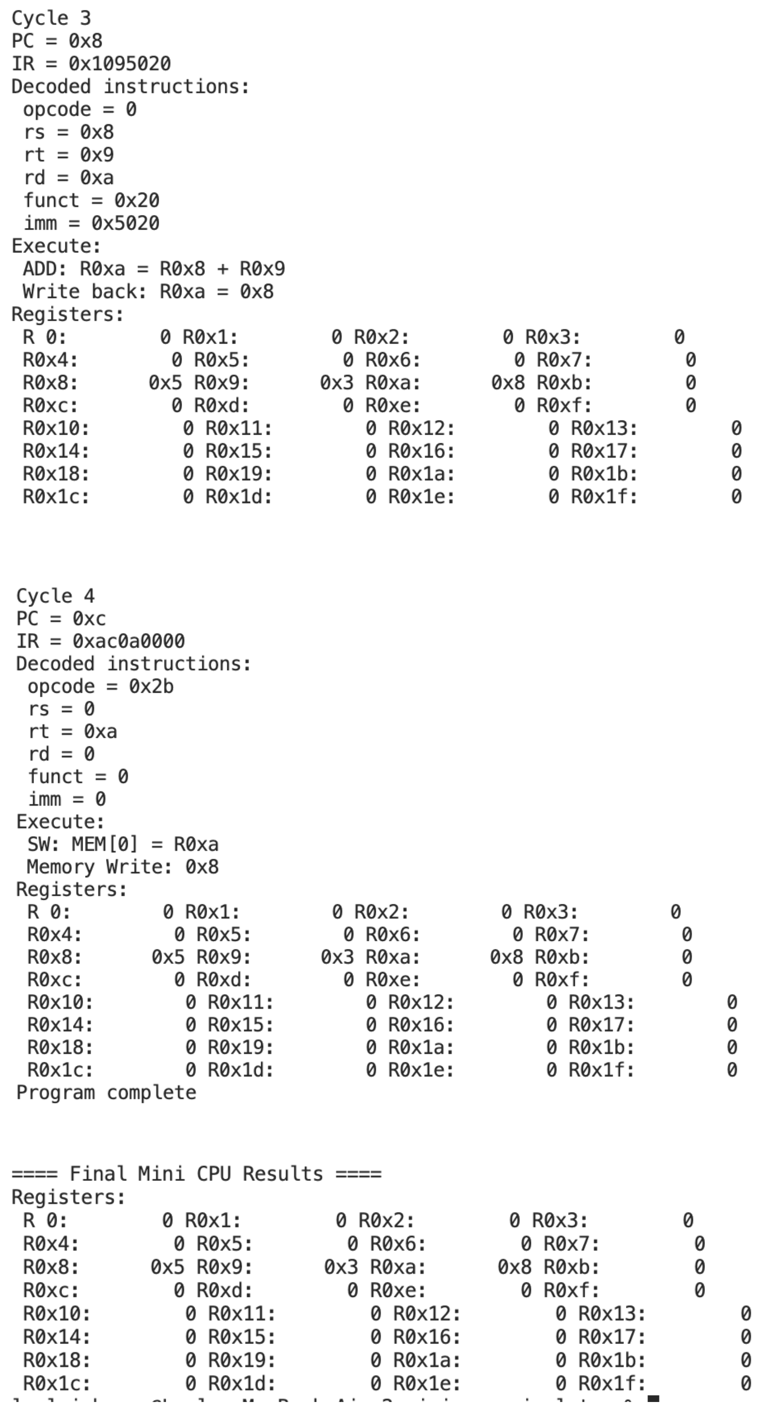
```
20080005
20090003
01095020
AC0A0000
```

Execution Trace

The following screenshots show the output of the test program:

```
==============================
Cycle 1
PC = 0
IR = 0x20080005
Decoded instructions:
 opcode = 0x8
 rs = 0
 rt = 0x8
 rd = 0
 funct = 0x5
 imm = 0x5
Execute:
 ADDI: R0x8 = R0 + 0x5
 Write back: R0x8 = 0x5
Registers:
 R 0:        0 R0x1:        0 R0x2:        0 R0x3:        0
 R0x4:        0 R0x5:        0 R0x6:        0 R0x7:        0
 R0x8:      0x5 R0x9:        0 R0xa:        0 R0xb:        0
 R0xc:        0 R0xd:        0 R0xe:        0 R0xf:        0
 R0x10:        0 R0x11:        0 R0x12:        0 R0x13:        0
 R0x14:        0 R0x15:        0 R0x16:        0 R0x17:        0
 R0x18:        0 R0x19:        0 R0x1a:        0 R0x1b:        0
 R0x1c:        0 R0x1d:        0 R0x1e:        0 R0x1f:        0


Cycle 2
PC = 0x4
IR = 0x20090003
Decoded instructions:
 opcode = 0x8
 rs = 0
 rt = 0x9
 rd = 0
 funct = 0x3
 imm = 0x3
Execute:
 ADDI: R0x9 = R0 + 0x3
 Write back: R0x9 = 0x3
Registers:
 R 0:        0 R0x1:        0 R0x2:        0 R0x3:        0
 R0x4:        0 R0x5:        0 R0x6:        0 R0x7:        0
 R0x8:      0x5 R0x9:      0x3 R0xa:        0 R0xb:        0
 R0xc:        0 R0xd:        0 R0xe:        0 R0xf:        0
 R0x10:        0 R0x11:        0 R0x12:        0 R0x13:        0
 R0x14:        0 R0x15:        0 R0x16:        0 R0x17:        0
 R0x18:        0 R0x19:        0 R0x1a:        0 R0x1b:        0
 R0x1c:        0 R0x1d:        0 R0x1e:        0 R0x1f:        0
```



```
Cycle 3
PC = 0x8
IR = 0x1095020
Decoded instructions:
 opcode = 0
 rs = 0x8
 rt = 0x9
 rd = 0xa
 funct = 0x20
 imm = 0x5020
Execute:
 ADD: R0xa = R0x8 + R0x9
 Write back: R0xa = 0x8
Registers:
 R 0:        0 R0x1:        0 R0x2:        0 R0x3:        0
 R0x4:        0 R0x5:        0 R0x6:        0 R0x7:        0
 R0x8:      0x5 R0x9:      0x3 R0xa:      0x8 R0xb:        0
 R0xc:        0 R0xd:        0 R0xe:        0 R0xf:        0
 R0x10:        0 R0x11:        0 R0x12:        0 R0x13:        0
 R0x14:        0 R0x15:        0 R0x16:        0 R0x17:        0
 R0x18:        0 R0x19:        0 R0x1a:        0 R0x1b:        0
 R0x1c:        0 R0x1d:        0 R0x1e:        0 R0x1f:        0


Cycle 4
PC = 0xc
IR = 0xac0a0000
Decoded instructions:
 opcode = 0x2b
 rs = 0
 rt = 0xa
 rd = 0
 funct = 0
 imm = 0
Execute:
 SW: MEM[0] = R0xa
 Memory Write: 0x8
Registers:
 R 0:        0 R0x1:        0 R0x2:        0 R0x3:        0
 R0x4:        0 R0x5:        0 R0x6:        0 R0x7:        0
 R0x8:      0x5 R0x9:      0x3 R0xa:      0x8 R0xb:        0
 R0xc:        0 R0xd:        0 R0xe:        0 R0xf:        0
 R0x10:        0 R0x11:        0 R0x12:        0 R0x13:        0
 R0x14:        0 R0x15:        0 R0x16:        0 R0x17:        0
 R0x18:        0 R0x19:        0 R0x1a:        0 R0x1b:        0
 R0x1c:        0 R0x1d:        0 R0x1e:        0 R0x1f:        0
Program complete


==== Final Mini CPU Results ====
Registers:
 R 0:        0 R0x1:        0 R0x2:        0 R0x3:        0
 R0x4:        0 R0x5:        0 R0x6:        0 R0x7:        0
 R0x8:      0x5 R0x9:      0x3 R0xa:      0x8 R0xb:        0
 R0xc:        0 R0xd:        0 R0xe:        0 R0xf:        0
 R0x10:        0 R0x11:        0 R0x12:        0 R0x13:        0
 R0x14:        0 R0x15:        0 R0x16:        0 R0x17:        0
 R0x18:        0 R0x19:        0 R0x1a:        0 R0x1b:        0
 R0x1c:        0 R0x1d:        0 R0x1e:        0 R0x1f:        0
```

The architecture of the program includes:

- 4 KB memory
- 32 general purpose registers
- Program Counter
- Instruction Register
- Execution model:
    - Fetch
    - Decode
    - Execute
    - Memory
    - Write Back