Instruction Set Architecture (ISA) - Part V

CS6133 - Computer Architecture I

Vikram Padman

NYU Polytechnic School of Engineering

vikram@poly.edu

Agenda

1. **Introduction** - Part I
2. **Classifying ISA** - Part I, II, III and IV
3. **Simply CPU - Revisited - Part V**
   - ISA - Instruction Set and encoding
   - Control - Main and ALU control units
   - Data Path - R-Type, I-Type and Branch
4. **Midterm - Project**

Instruction Set and Encoding

- **Instructions:**
  - **Computational** - add, sub, or, and: rd = rs *op* rt, R-Type & addi: rt=rs+IMM, I-Type
  - **Memory** - lw, sw: Load word, rt=mem[rs+offset], store word, mem[rs+offset]=rt, I-Type
  - **Compare** - slt: Set on Less than, rd=(rs < rt)?1:0, R-Type
  - **Control** - beq: Branch equal, pc += (rs==rt)?(Addr+4):4, I-Type
  - **I/O** - out: Output, out port = rs, O-Type
- **Memory Alignment :** 32-bit word aligned
- **Number of Registers :** 32
- **Format :** 32-bit Fixed format

R-Type Instruction

Opcode (31:26)	rs (25:21)	rt (20:16)	rd (15:11)	shamt (10:6)	funct (5:0)

I-Type Instruction

Opcode (31:26)	rs (25:21)	rt (20:16)	address or Immediate (15:0)

Branch Instruction

Opcode (31:26)	rs (25:21)	rt (20:16)	address (15:0)

O-Type Instruction

Opcode (31:26)	rs (25:21)	XXXX - Don't Care (20:0)

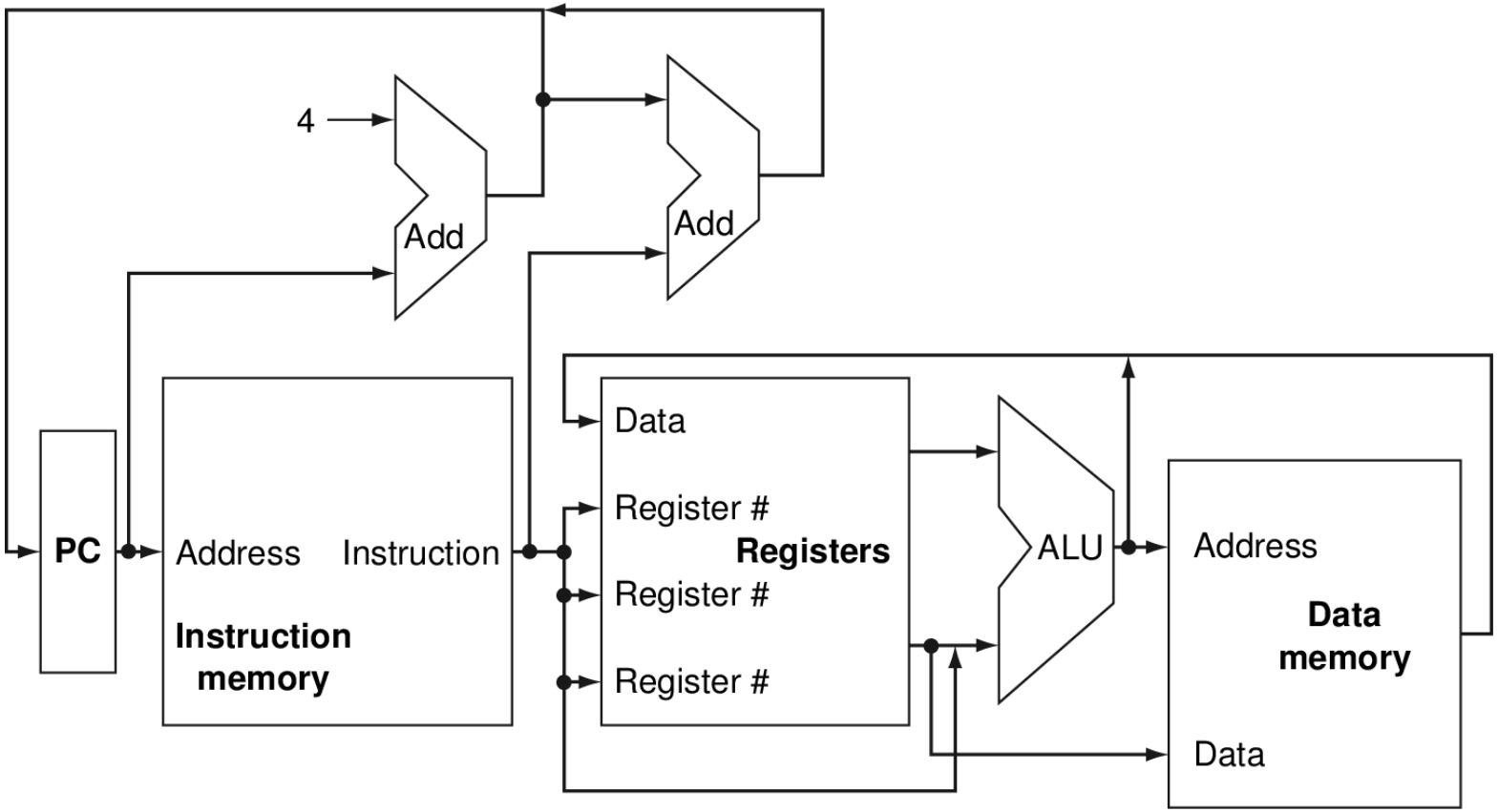
Instruction Set and Encoding

1. **add :** Opcode = b"000000", funct = b"100000"
2. **sub :** Opcode = b"000000", funct = b"100010"
3. **and :** Opcode = b"000000", funct = b"100100"
4. **or :** Opcode = b"000000", funct = b"100010"
5. **addi :** Opcode = b"100000", funct = b"XXXXXX"[1]
6. **nop :** Opcode = b"000000", funct = b"000000"
7. **slt :** Opcode = b"000000", funct = b"101010"
8. **lw :** Opcode = b"100011", funct = b"XXXXXX"[1]
9. **sw :** Opcode = b"101011", funct = b"XXXXXX"[1]
10. **beq :** Opcode = b"000100", funct = b"XXXXXX"[1]
11. **out :** Opcode = b"101100", funct = b"XXXXXX"[1]

[1] X = Don't care

Need for Control signals

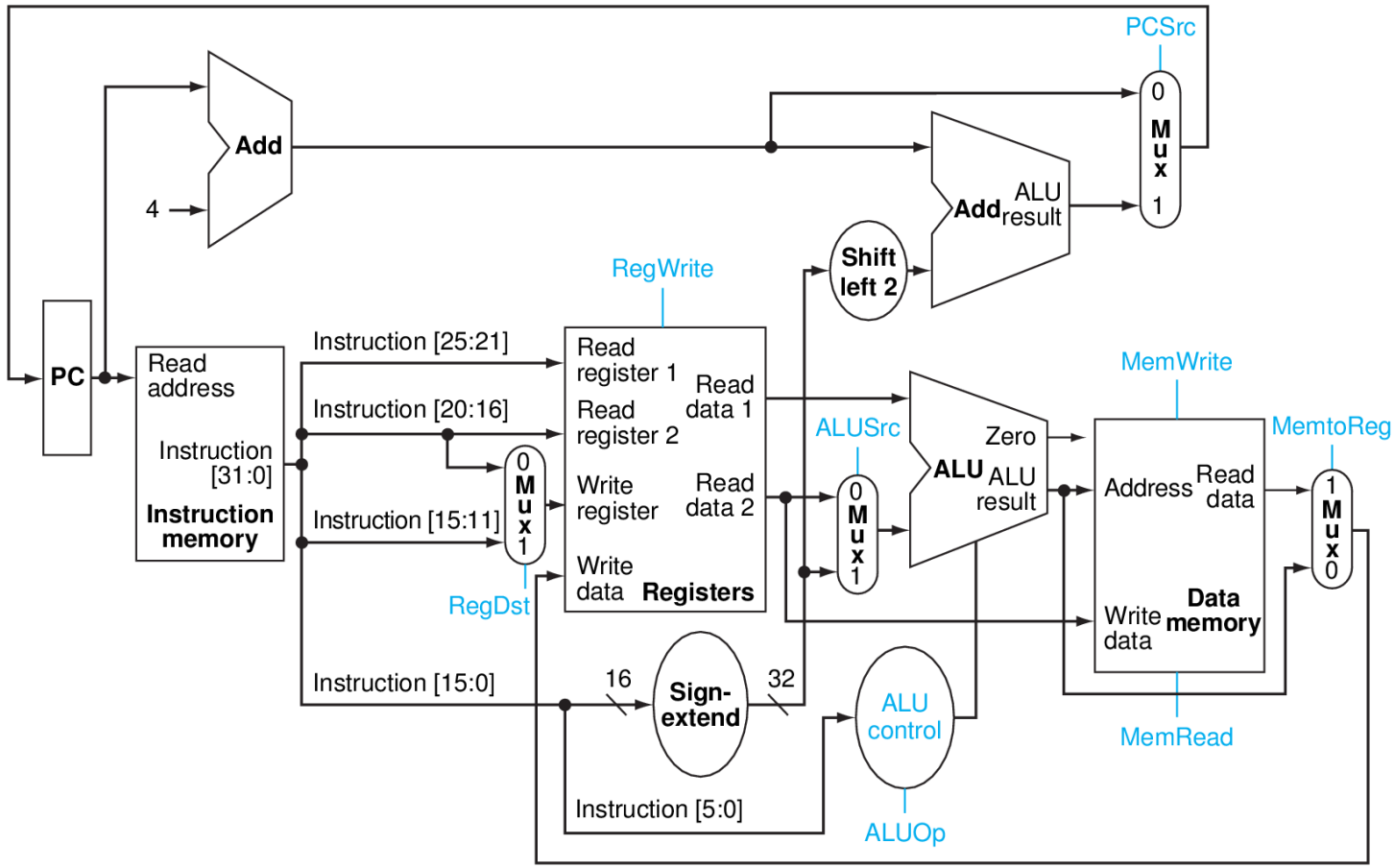
Simple CPU's top level:



From "Computer Organization and Design" page 302

Need for Control signals

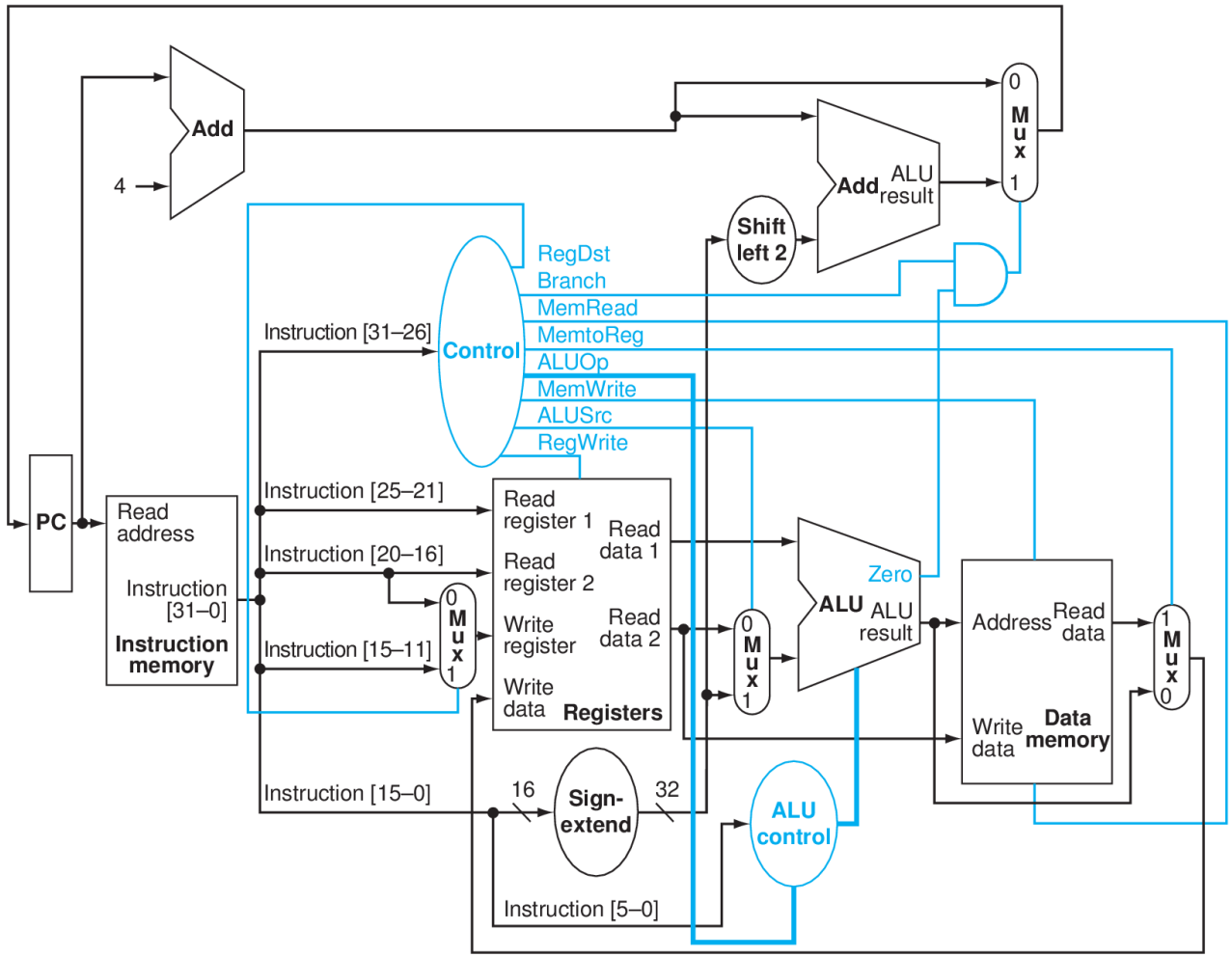
Simple CPU's top level with MUXs and control signals:



From "Computer Organization and Design" page 320

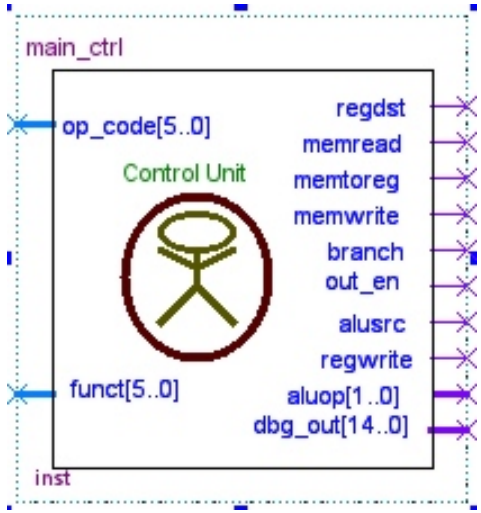
Need for Control signals

Simple CPU's top level with Main and ALU control units



From "Computer Organization and Design" page 322

Main Control Unit



Opcode	RegDest	MemToReg	MemRead	MemWrite	ALUOp	ALUSrc	RegWrite	Branch	OutEn
lw:b"100011"	b'0'	b'1'	b'1'	b'0'	b"00"	b'1'	b'1'	b'0'	b'0'
sw:b"101011"	b'X'	b'X'	b'0'	b'1'	b"00"	b'1'	b'0'	b'0'	b'0'
beq:b"000100"	b'X'	b'X'	b'0'	b'0'	b"01"	b'0'	b'0'	b'1'	b'0'
add:b"000000"	b'1'	b'0'	b'0'	b'0'	b"10"	b'0'	b'1'	b'0'	b'0'
sub:b"000000"	b'1'	b'0'	b'0'	b'0'	b"10"	b'0'	b'1'	b'0'	b'0'
and:b"000000"	b'1'	b'0'	b'0'	b'0'	b"10"	b'0'	b'1'	b'0'	b'0'
or:b"000000"	b'1'	b'0'	b'0'	b'0'	b"10"	b'0'	b'1'	b'0'	b'0'
slt:b"000000"	b'1'	b'0'	b'0'	b'0'	b"10"	b'0'	b'1'	b'0'	b'0'
nop:b"000000"	b'X'	b'X'	b'X'	b'0'	b"XX"	b'X'	b'0'	b'0'	b'0'
addi:b"100000"	b'0'	b'0'	b'1'	b'0'	b"00"	b'1'	b'1'	b'0'	b'0'
out:b"000000"	b'X'	b'X'	b'X'	b'0'	b"XX"	b'X'	b'0'	b'0'	b'1'

Main Control Unit

Instruction Set Architecture (ISA) - Part V

Vikram Padman

Agenda

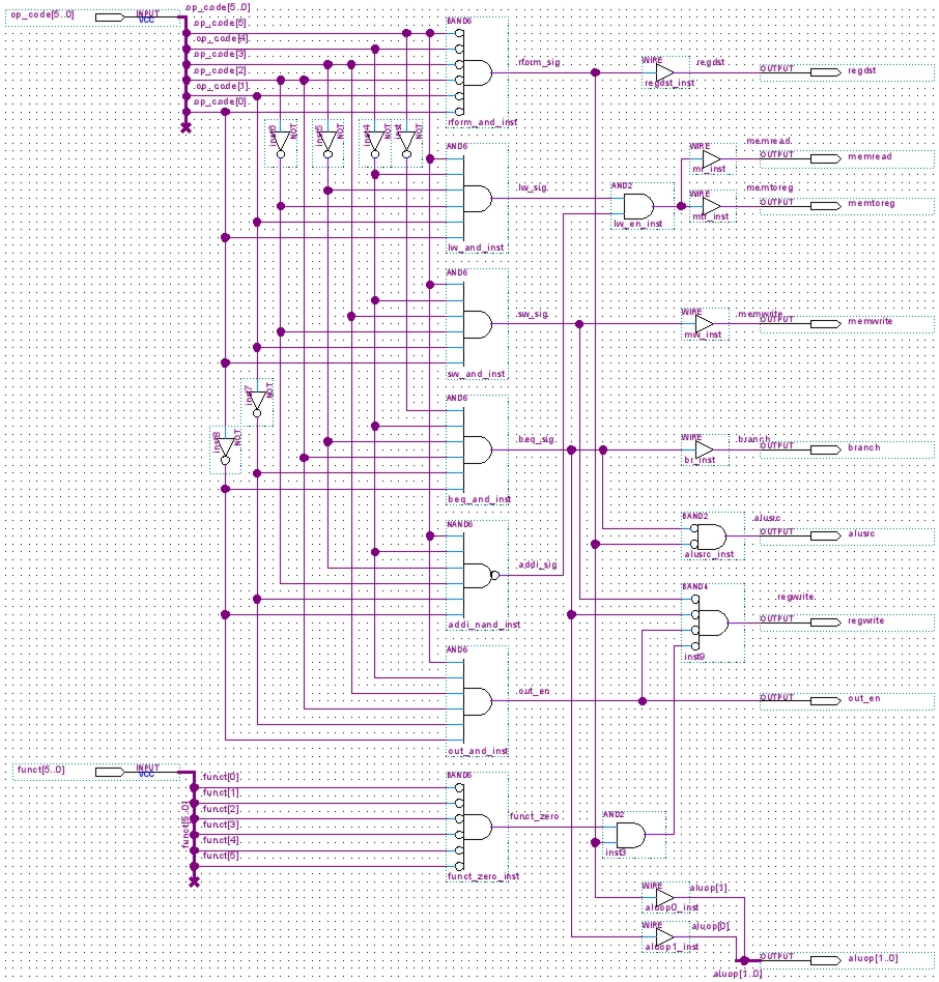
Simply CPU

ISA

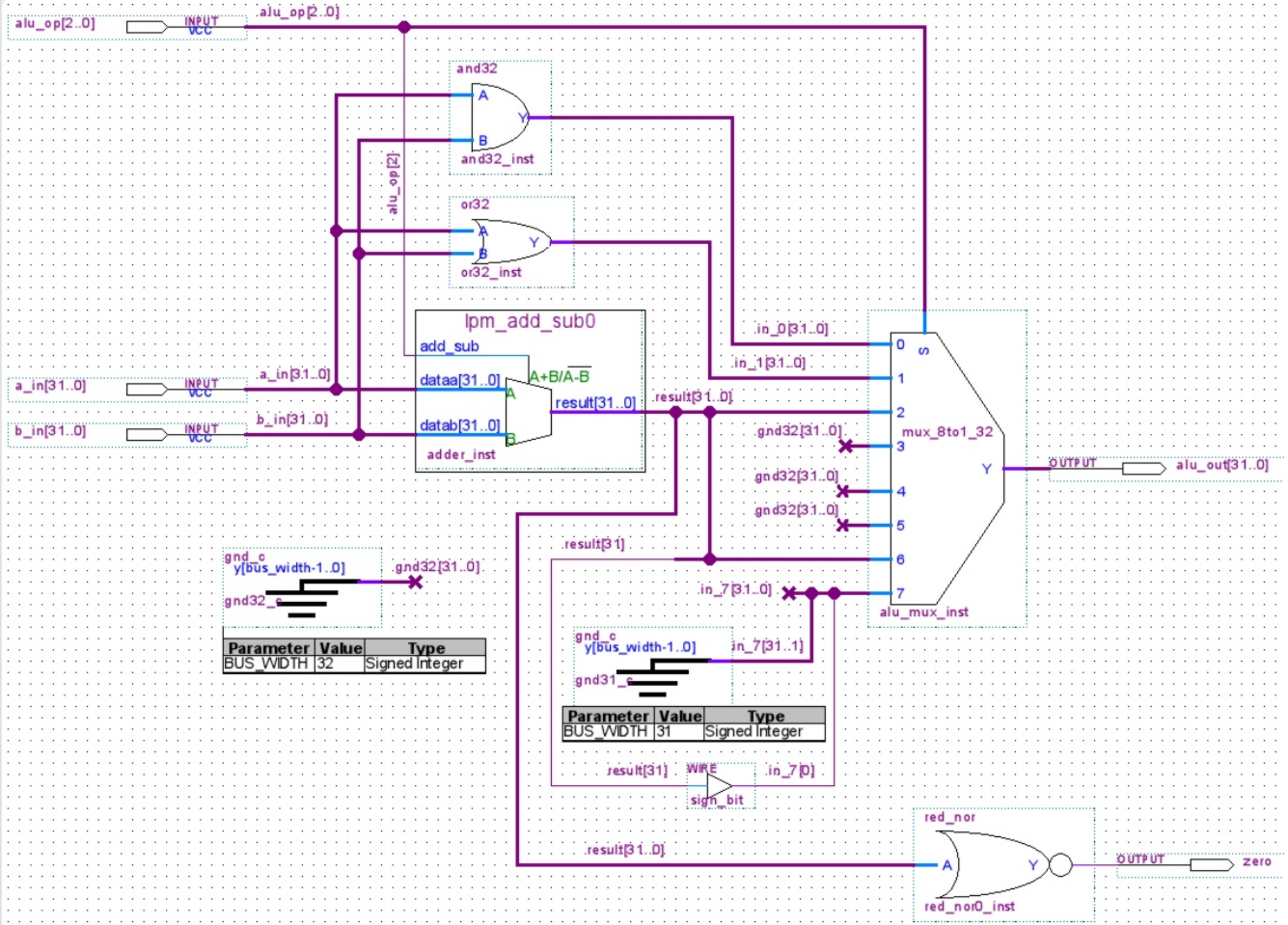
Control

Data Path

Mid Term

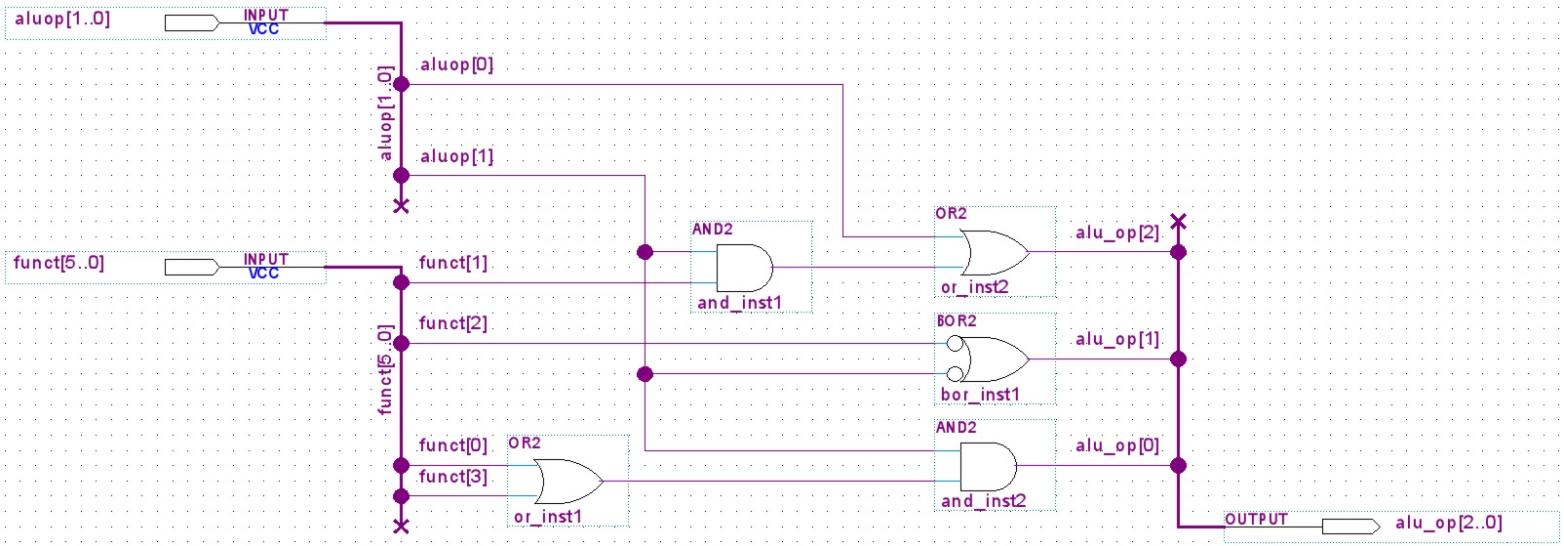


ALU Functions



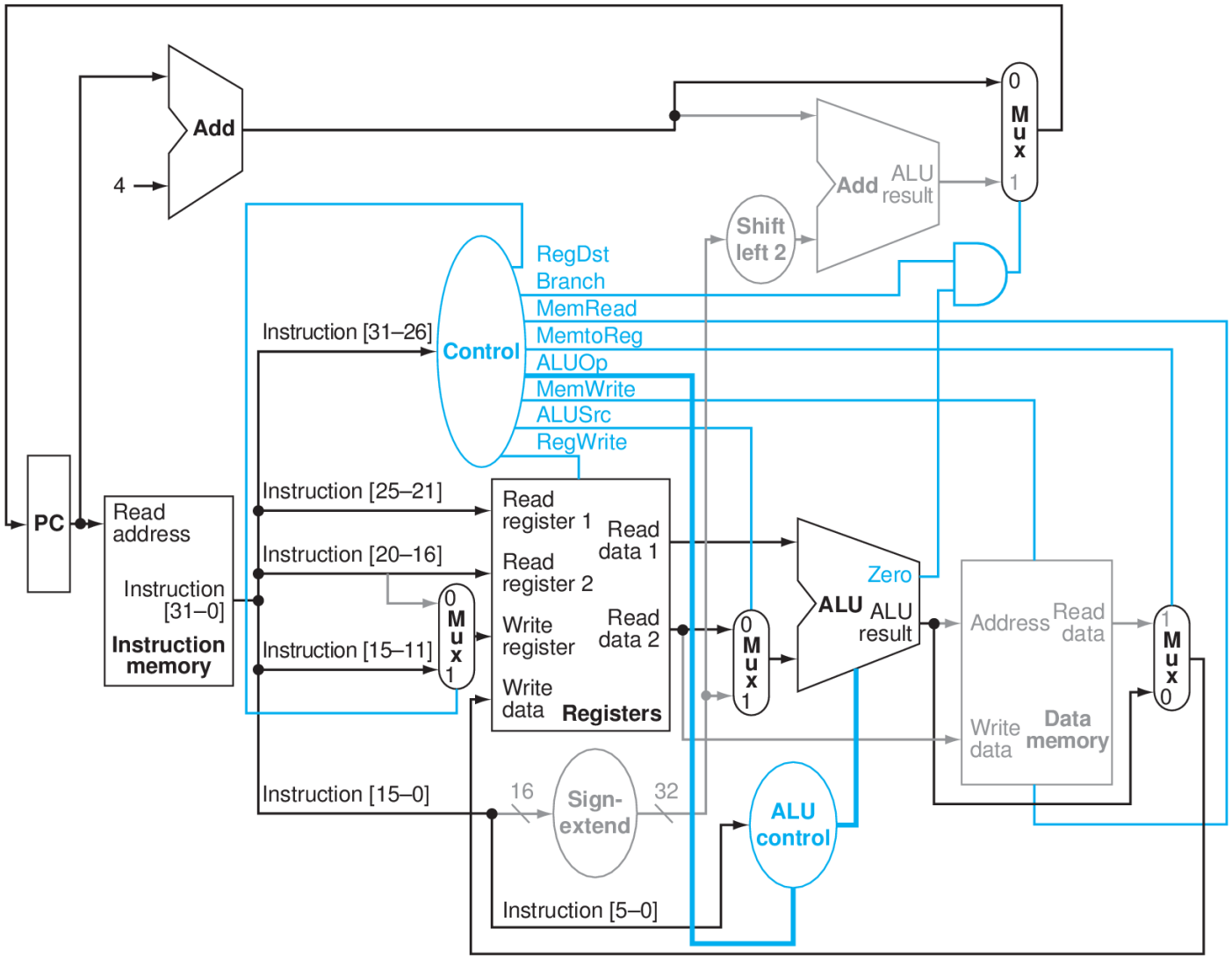
ALUop	Function
b“000”	AND
b“001”	OR
b“010”	add
b“110”	sub
b“111”	slt
Others	nop

ALU Control Unit



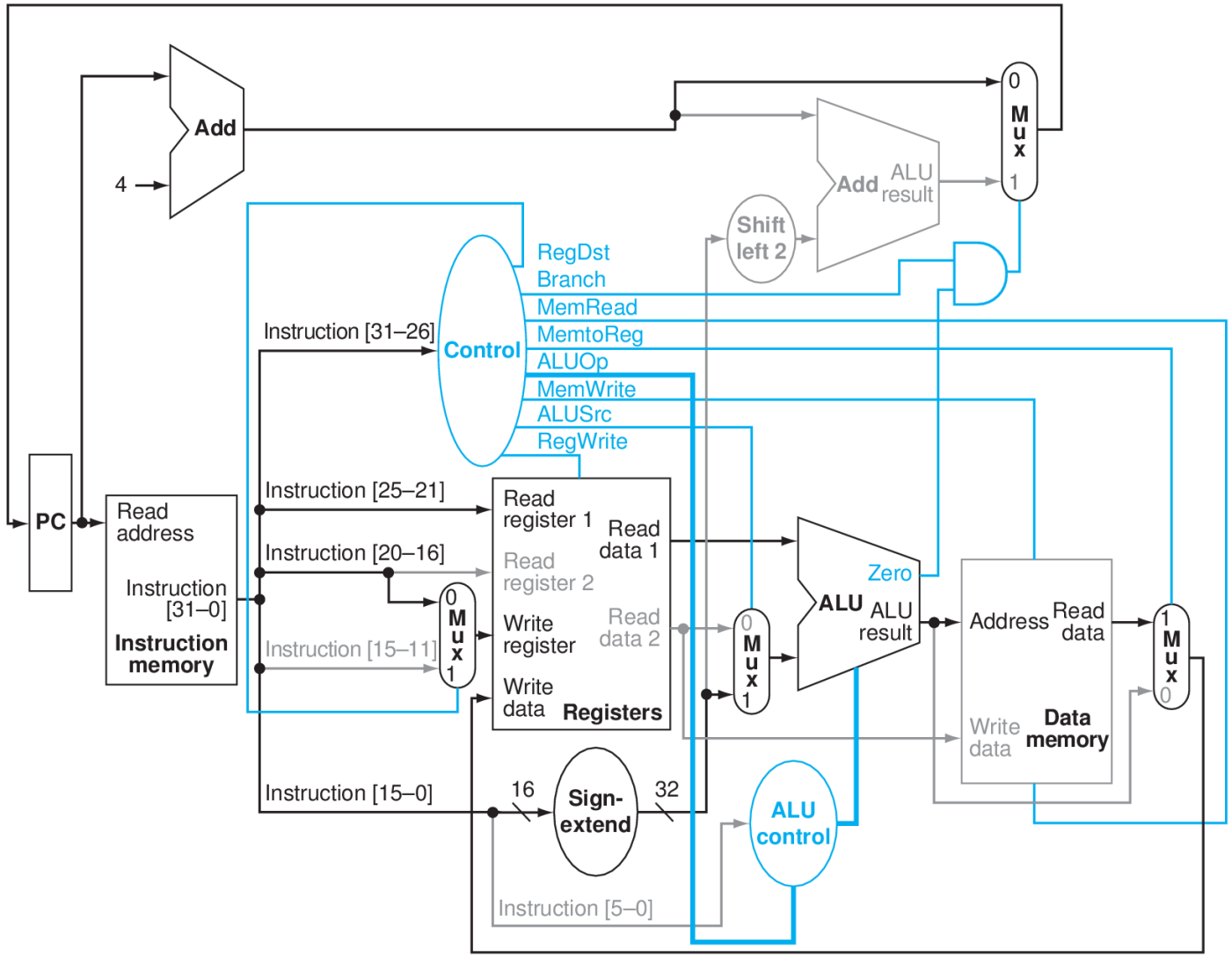
Opcode	aluop	funct	ALU Func.	alu_op
lw:b"100011"	b"00"	XXXXXX	add	b"010"
sw:b"101011"	b"00"	XXXXXX	add	b"010"
beq:b"000100"	b"01"	XXXXXX	sub	b"110"
add:b"000000"	b"10"	b"100000"	add	b"010"
sub:b"000000"	b"10"	b"100010"	sub	b"110"
and:b"000000"	b"10"	b"100100"	and	b"000"
or:b"000000"	b"10"	b"100101"	or	b"001"
slt:b"000000"	b"10"	b"101010"	slt	b"111"

R-Type Instruction



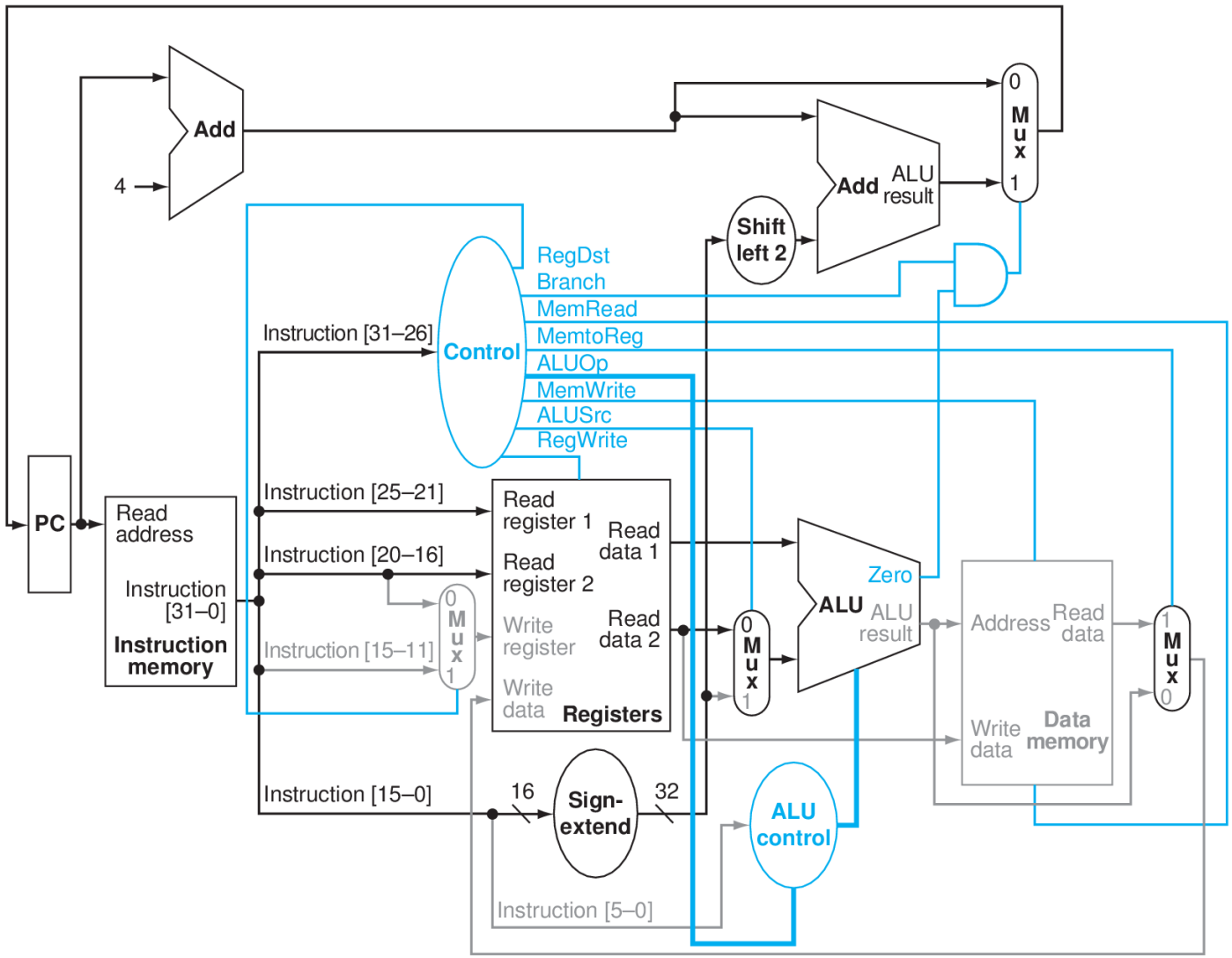
From "Computer Organization and Design" page 324

I-Type Instruction



From "Computer Organization and Design" page 324

Branch Instruction



From "Computer Organization and Design" page 324

Simple CPU in DE0-Nano

Instruction Set Architecture (ISA) - Part V

Vikram Padman

Agenda

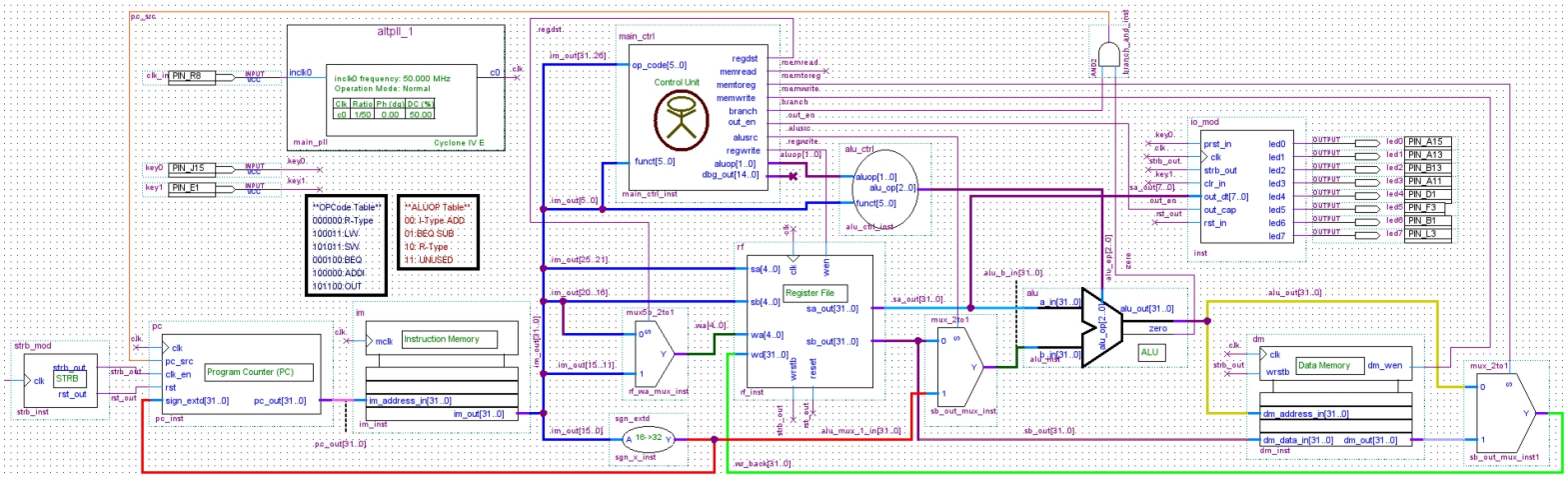
Simply CPU

ISA

Control

Data Path

Mid Term



Mid Term Project

Part 1: MIPS Registers (10 Points)

1. The data memory in MIPS CPU is running at the same speed as the CPU so it could fetch data in one clock cycle, just like the register file (RF).
    1. what is purpose of having a 32 registers? Could RF be reduced to 16 or 8? (2.5)
    2. What is impact and benefits of reducing the size of the RF? (2.5)
2. Why is instruction memory read only? Are there any advantages in enabling write access to instruction memory? (5)

Mid Term Project

Part 2: Understanding Simple CPU (mips_ss_v2.qar) (20 Points)

Understanding Simple CPU (mips_ss_v2) is important to complete the rest of the midterm project. In this part you will write an assembly programs in binary and execute it in mips_ss_v2 CPU running in DE0-Nano.

Fibonacci Number Generator: (20)

1. Write a program that generates Fibonacci numbers up to Fib-40 and could backwards to 0 decrementing by 1.
2. Your program should display 8 least significant bits of each number you generate in the above part through the LEDs.
3. While counting back you should blink the LEDs whenever the count value is equal to a Fibonacci number.

For this part you must submit the programs you wrote and signal tap II capture file.

Mid Term Project

Part 3: Advanced Addressing modes (mips_ss_v2.qar) (70 Points)

As implemented, mips_ss_v2 only supports register and immediate addressing modes. For this part add the following addressing modes to mips_22_v2:

1. Indexed (30)
2. Autoincrement and Autodecrement (40)

You report should at least contain, but not limited to, the following: Instruction formats, implementation details, modification to control unit and signal, detailed description of any new modules you implemented.[1]

Refer to the last years midterm project report to understand what your report should contain.

[1] Partial credit (75% max) will awarded for a reasonable non-functional