CE 1911

Single Cycle Processor III

Simple Data Path

- Arithmetic and logical

fn Reg1, Reg2, Wreg	Wreg ← Reg1 fn Reg2

- Memory

ld Reg1, Wreg	Wreg ← MEM(Reg1)
st Reg1, Reg2	MEM(Reg1) ← Reg2

- Immediate

ld Reg1, “imm value”	Reg1 ← “imm value”

Simple Data Path

- Arithmetic and logical

sub RA, RB, RC

RC ← RA - RB

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0

RA RB RC

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100

00 – A
01 – B
10 – C
11 – D

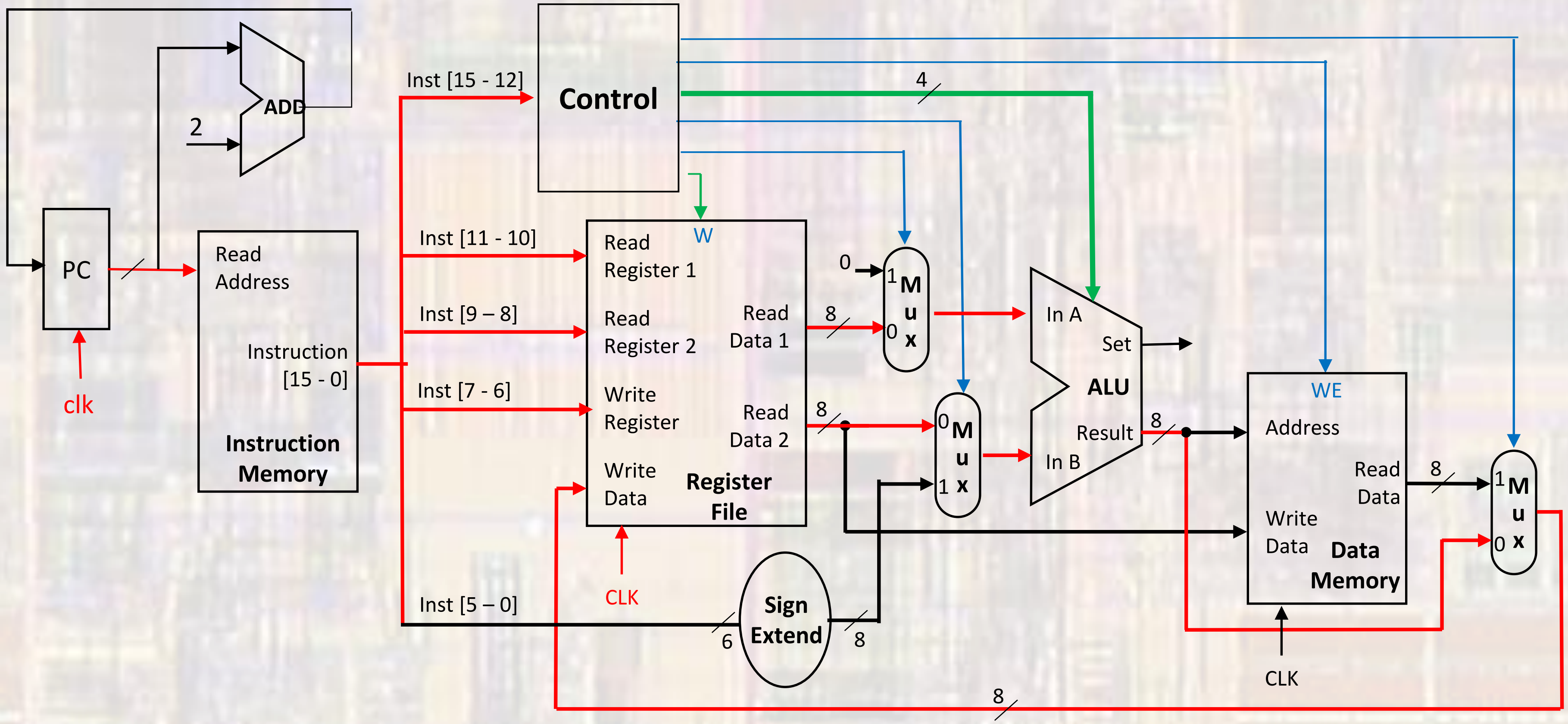
Technically these are don't care but we will always code them as 0s

Simple Data Path

sub RA, RB, RC

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0

- Operation
  - Arithmetic and Logical



Simple Data Path

- Instruction Format

  - Instruction Encoding
    - LD WRreg ← MEM(Reg1)
    - LD RA, RC RC ← MEM(RA)

Instruction				Reg 1		Reg 2		WR Reg		Immediate Value					
1	0	0	0	0	0	x	x	1	0	0	0	0	0	0	0

RA RC

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100

00	A
01	B
10	C
11	D

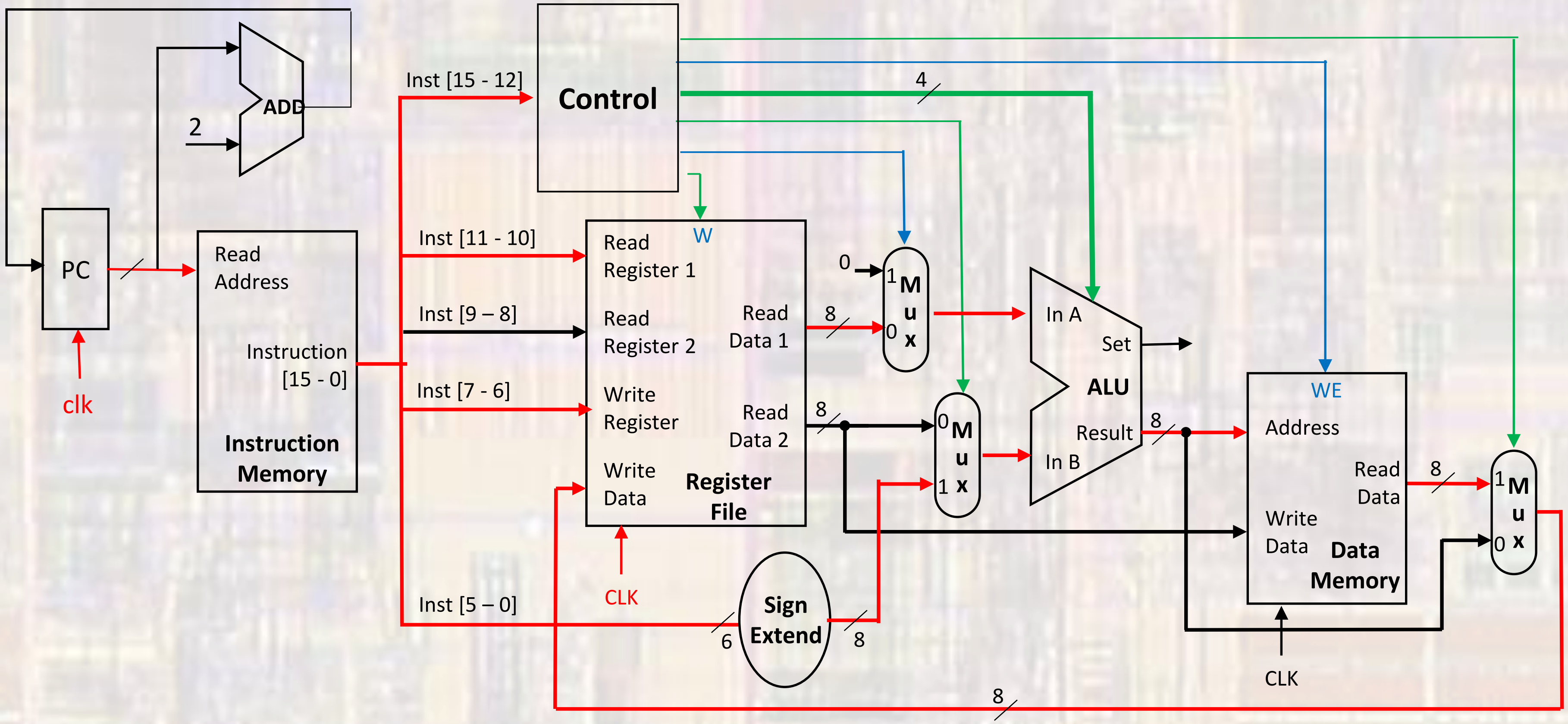
added to Reg 1 to make the memory address

Simple Data Path

- Operation
  - LD

LD RA, RC

Instruction				Reg 1		Reg 2		WR Reg		Immediate Value					
1	0	0	0	0	0	x	x	1	0	0	0	0	0	0	0



Simple Data Path

- Instruction Format

- Instruction Encoding
  - ST   Reg1, Reg2   MEM(Reg1) ← Reg2
  - ST   RA, RC   MEM(RA) ← RC

Instruction				Reg 1		Reg 2		WR Reg		Immediate Value					
1	0	0	1	0	0	1	0	x	x	0	0	0	0	0	0

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100

RA

RC

00	A
01	B
10	C
11	D

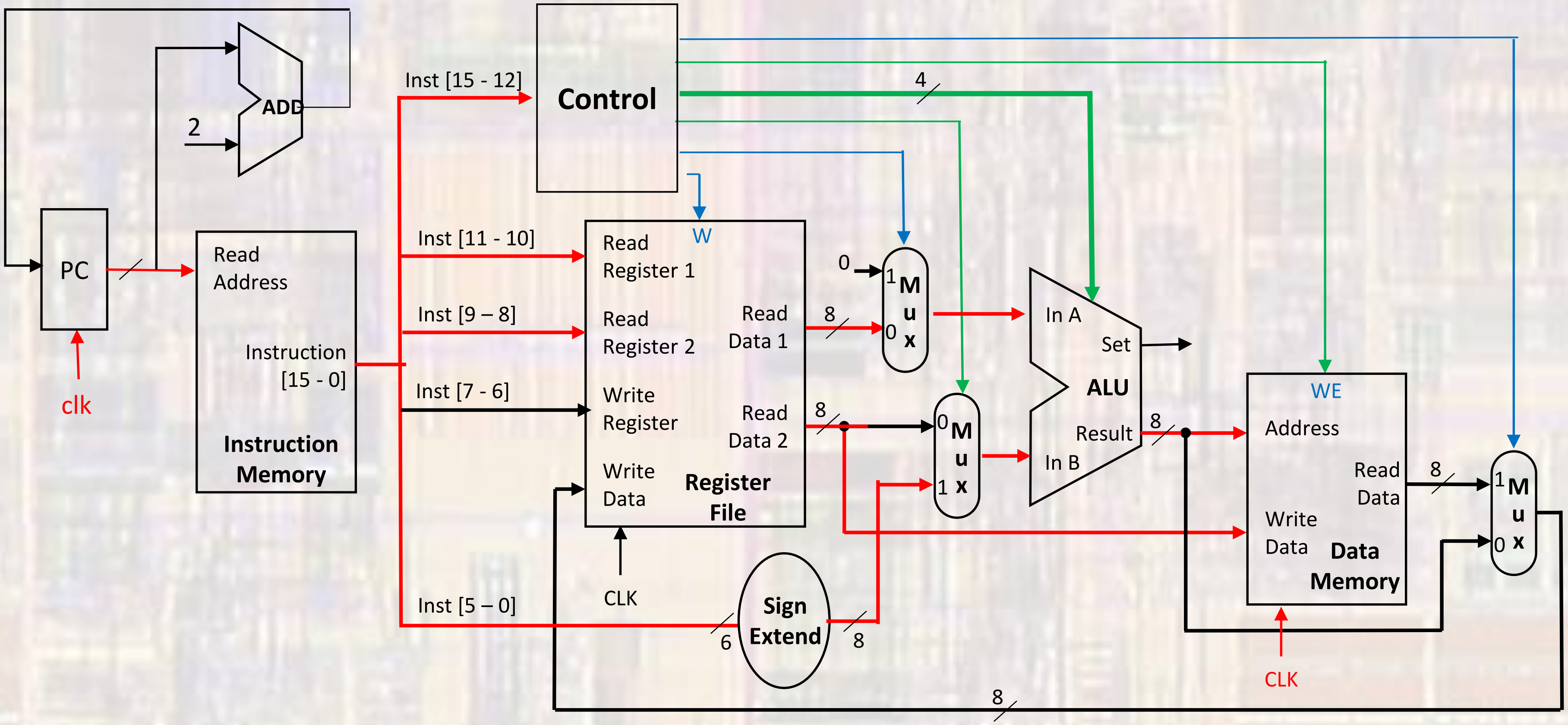
added with Reg 1 to make the memory address

Simple Data Path

- Operation
  - ST

ST RA, RC

Instruction				Reg 1		Reg 2		WR Reg		Immediate Value					
1	0	0	1	0	0	1	0	x	x	0	0	0	0	0	0



Simple Data Path

- Instruction Format

  - Instruction Encoding
    - LDI   WrRreg ← Imm extended
    - LDI   RA, 0x12   RA ← 0x12

Instruction				Reg 1		Reg 2		WR Reg		Immediate Value					
1	1	0	0	x	x	x	x	0	0	0	1	0	0	1	0

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100

RA

00 – A
01 – B
10 – C
11 – D

signed Hex
0x20 to 0x1F

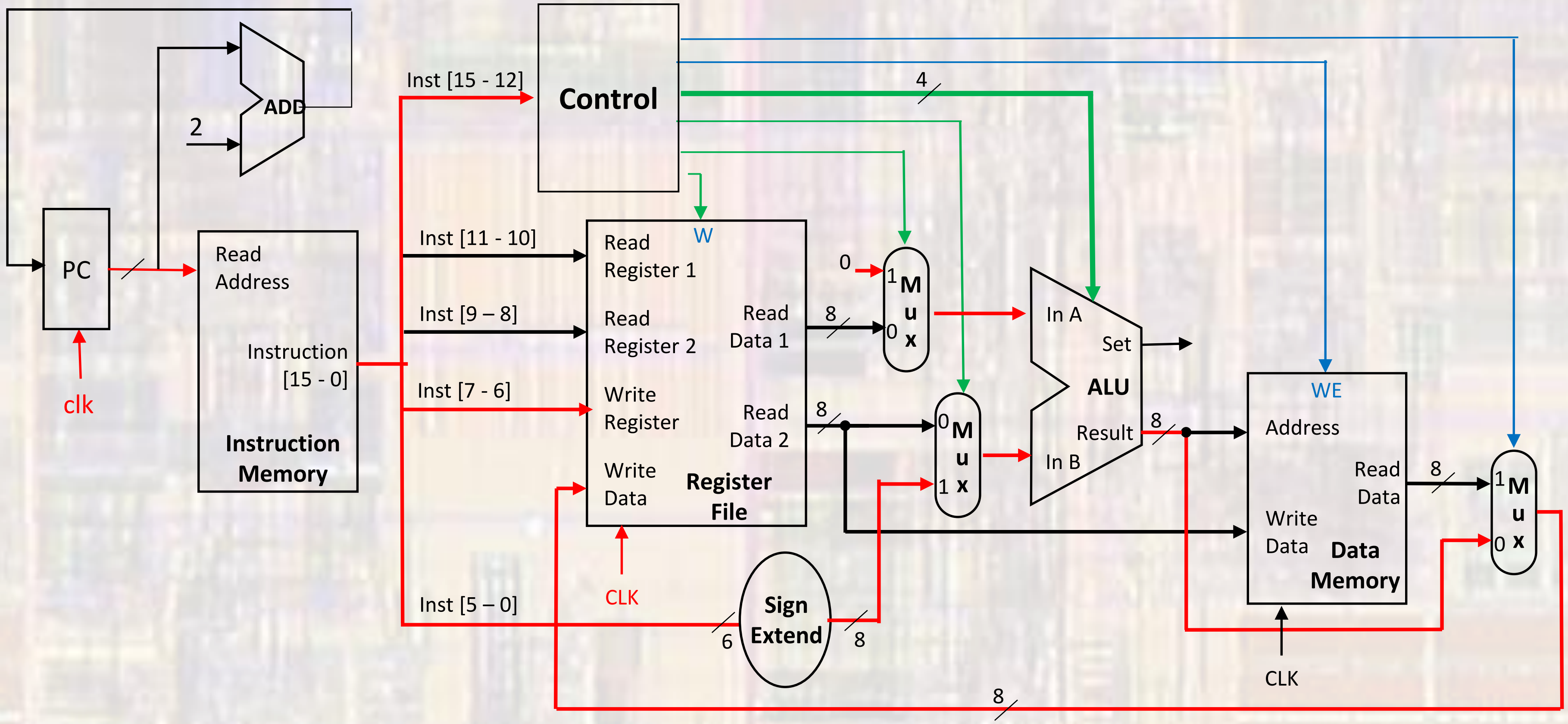
100000 to 011111
-32 to 31

Simple Data Path

- Operation
  - LDI

LDI RA, 0x12

Instruction				Reg 1		Reg 2		WR Reg		Immediate Value					
1	1	0	0	x	x	x	x	0	0	0	1	0	0	1	0



Simple Data Path

- ALU Control
  - Basic ALU control mapping

Operation	invA	negB	ctl[1]	ctl[0]
AND	0	0	1	1
OR	0	0	1	0
NOR	1	1	1	1
NAND	1	1	1	0
ADD	0	0	0	1
SUB	0	1	0	1
SLT	0	1	0	0

Simple Data Path

- ALU Control

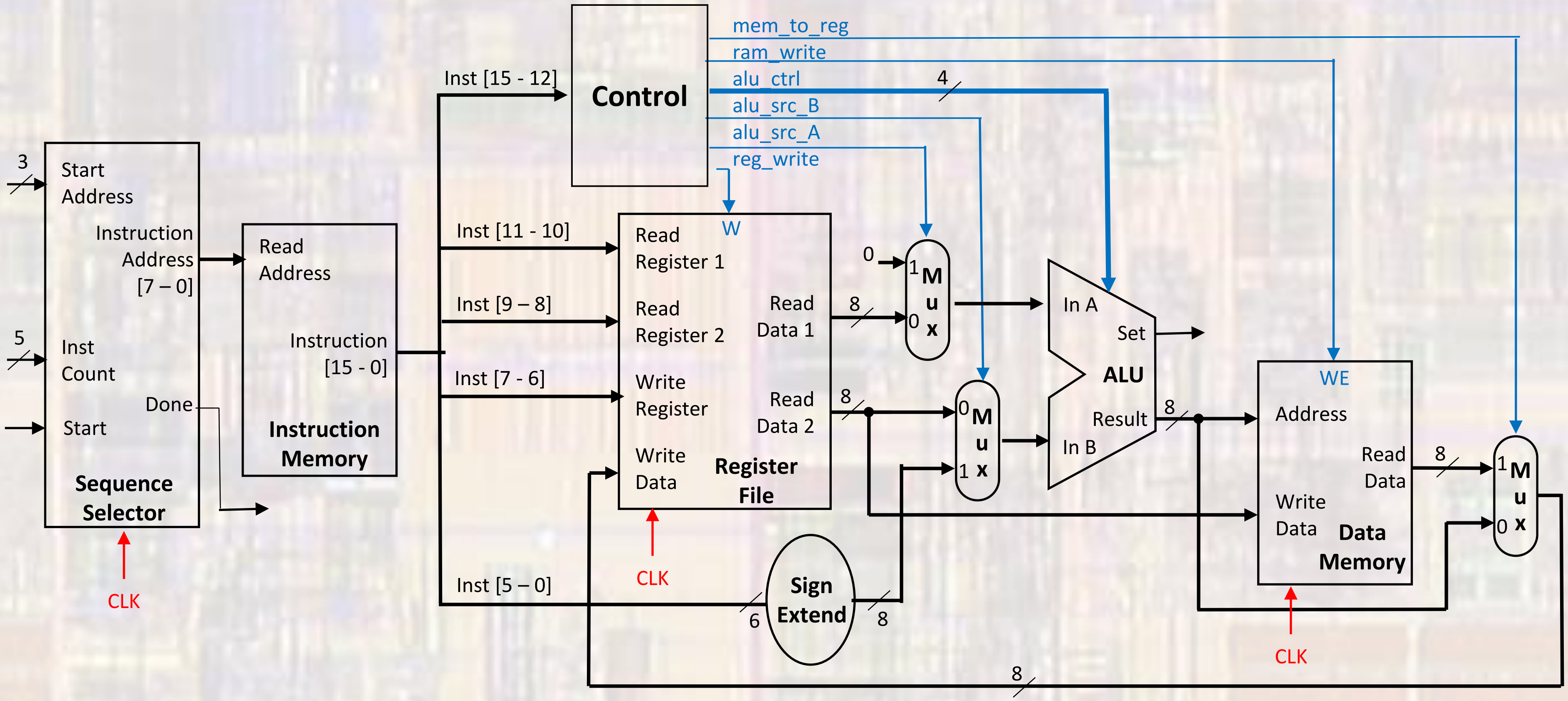
  - Full ALU control mapping

Operation	invA	negB	ctl[1]	ctl[0]
AND	0	0	1	1
OR	0	0	1	0
NOR	1	1	1	1
NAND	1	1	1	0
ADD	0	0	0	1
SUB	0	1	0	1
SLT	0	1	0	0
LD	0	0	0	1
ST	0	0	0	1
LDI	0	0	0	1

adding 0 from instruction
adding 0 from instruction
adding 0 from mux

Simple Data Path

- Additional Control Signals



Simple Data Path

- Additional Control Signals

  - mem_to_reg
    - Selects between the ALU output and the RAM read data value to pass to the Write data input of the register file
    - “1” only on LD instructions

  - ram_write
    - Enable writing to the Data RAM
    - “1” only on ST instructions

Simple Data Path

- Additional Control Signals

  - alu_src_B
    - Selects between the Read data 2 output of the register file and the sign extended immediate signal to pass to ALU input B
    - “0” for arithmetic and logical instructions
    - “1” for LD, ST and LDI instructions

  - alu_src_A
    - Selects between the Read data 1 output of the register file and 0 to pass to ALU input A
    - “1” only on LDI commands

Simple Data Path

- Additional Control Signals

  - reg_write
    - Enable writing to the Register File
    - "0" only on ST instructions

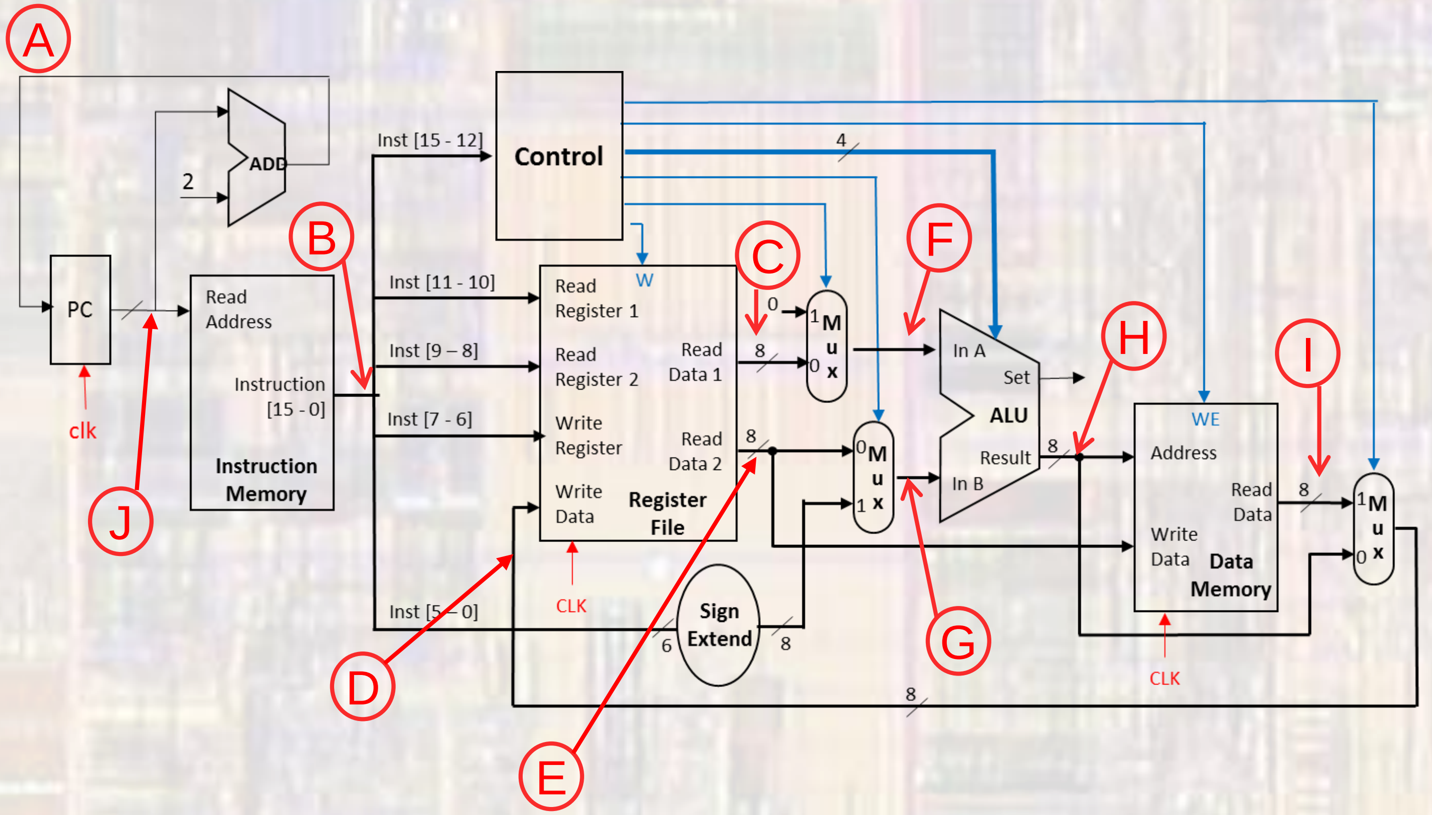
Simple Data Path

- Performance Issues
  - Longest delay determines clock period
    - Critical path: load instruction
    - Instruction memory → register file → ALU → data memory → register file
  - Not feasible to vary period for different instructions
  - Violates design principle
    - Making the common case fast
  - We will improve performance by pipelining

Simple Data Path

- Assume RA, RB, RC, RD contain the values 0x11, 0x22, 0x44, 0x88 respectively

Provide the values for each signal after executing the instruction: add RA,RB,RA located in program memory at 0x56



Node	Value (hex)
A	0x58
B	0x4100
C	0x11
D	0x33
E	0x22
F	0x11
G	0x22
H	0x33
I	??
J	0x56