Lecture 12: Single-Cycle Control Unit

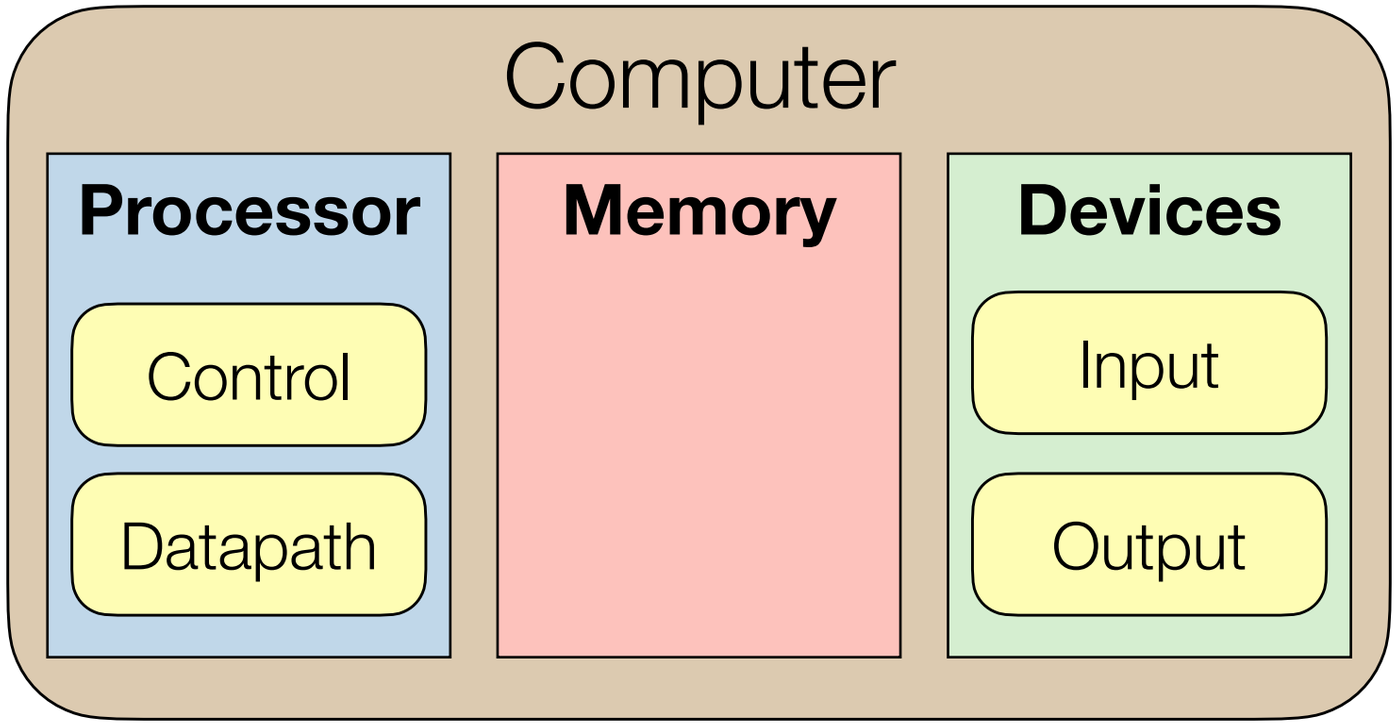
Spring 2024

Jason Tang

Topics

- Control unit design
- Single cycle processor
- Control unit circuit implementation

Computer Organization



1. Analyze instruction set

2. Select datapath components and clocking methodology

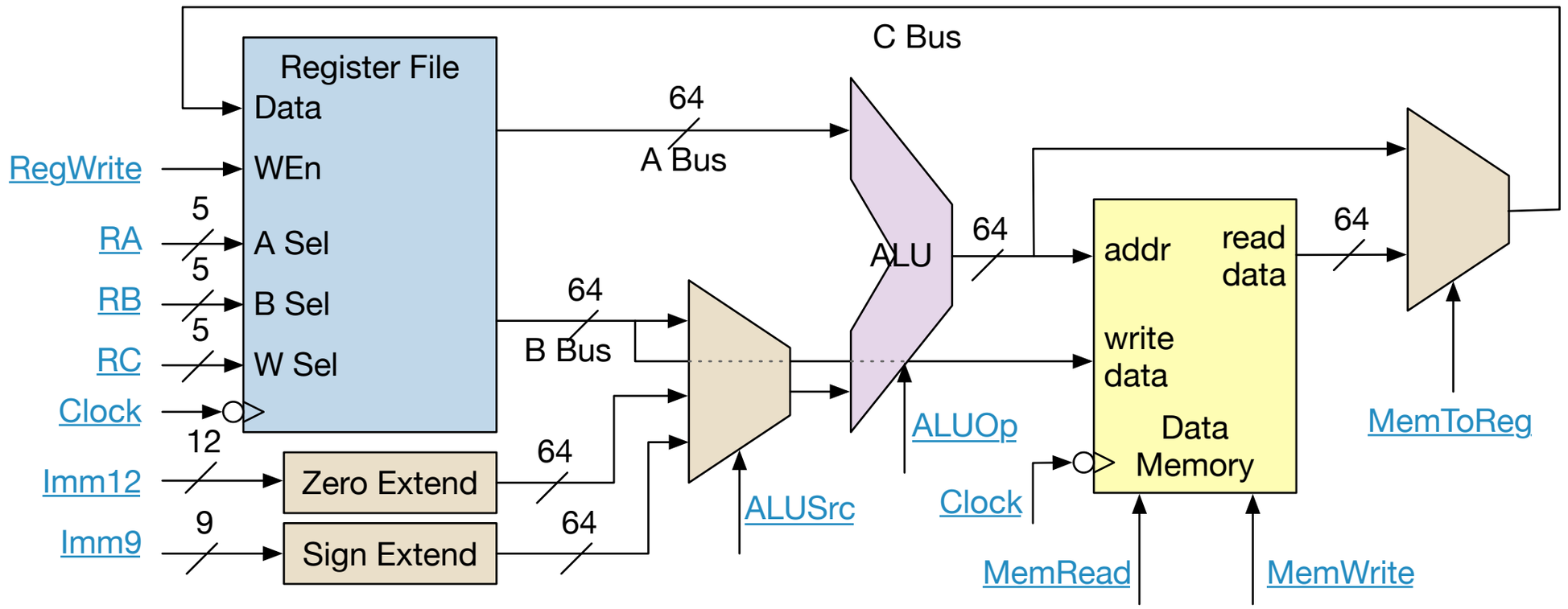
3. Assemble datapath

4. Analyze implementation of each instruction to determine control points

5. Assemble control logic

Single-Cycle Datapath

- Whereas last lecture described the components of the datapath, this lecture discusses how to generate the control signals (underlined in blue)



Control Unit

- Portion of the CPU that takes an instruction and determines which operation to perform (instruction decoding)
    - Given an instruction, determines which values to write to each control line
    - Determines count and type of operands (i.e., R-Type, D-Type, etc.)
    - Determines all dependencies along datapath
- Can be built via combinatorial logic or microcode

Example ARMv8-A Instructions

- First 11 bits of instruction largely determines instruction
  - Subset of those bits give the instruction type

Instruction	A64 Instruction Set Encoding											Inst Type	Section†
	31	30	29	28	27	26	25	24	23	22	21		
add (shifted reg)	1	0	0	0	1	0	1	1	x	x	0	R	C6.2.5
sub (shifted reg)	1	1	0	0	1	0	1	1	x	x	0	R	C6.2.358
add (immediate)	1	0	0	1	0	0	0	1	x	x	x	I	C6.2.4
and (immediate)	1	0	0	1	0	0	1	0	0	x	x	I	C6.2.12
ldur	1	x	1	1	1	0	0	0	0	1	0	D	C6.2.202
stur	1	x	1	1	1	0	0	0	0	0	0	D	C6.2.346

† Referenced section numbers are from the *ARM Architecture Reference Manual for A-profile architecture*

Setting ALUOp

- Depending upon instruction type, the ALU operation is encoded as two bits

- For R-Types, when bit 24 is 1 and bit 21 is 0, then bits 30-29 specifies whether to add (00) or subtract (10)

Instruction	A64 Instruction Set Encoding											Inst Type
	31	30	29	28	27	26	25	24	23	22	21	
add (shifted reg)	1	0	0	0	1	0	1	1	x	x	0	R
sub (shifted reg)	1	1	0	0	1	0	1	1	x	x	0	R
add (immediate)	1	0	0	1	0	0	0	1	x	x	x	I
and (immediate)	1	0	0	1	0	0	1	0	0	x	x	I

- For I-Types, when bits 25-23 give the type of processing (add/subtract, logical, etc) and then bits 30-29 give specific ALU operation

  - For processing type 01x, bits 30-29 specifies add (00) or subtract (10)

  - For processing type 100, bits 30-29 specifies bitwise ADD or OR

ARM Architecture Reference Manual for A-profile architecture, sections C4.1.86 and C4.1.89

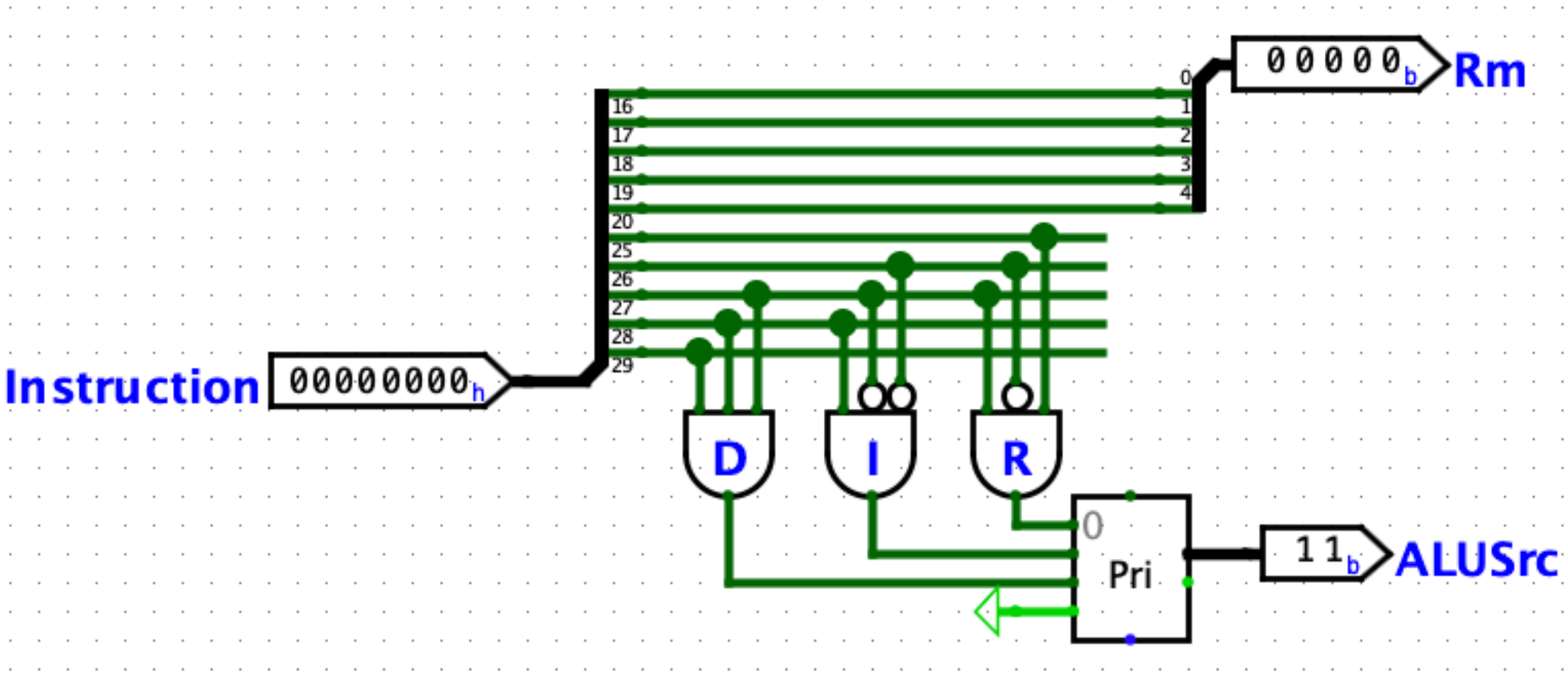
ARMv8-A Register Selects

- Control unit first determines instruction type to determine number and location of operands
  - Based upon operand type, it sets register select and ALUSrc controls
- Opcode is 6 to 11 bits wide and is in upper portion of each instruction
- Destination register, if there is one, is encoded as bits [4:0]
- First operand, if exists, precedes destination at bits [9:5]

R-Type	opcode	R_m	shamt	R_n	R_d
	11 bits	5 bits	6 bits	5 bits	5 bits
D-Type	opcode	address	op2	R_n	R_t
	11 bits	9 bits	2 bits	5 bits	5 bits
I-Type	opcode	immediate		R_n	R_d
	10 bits	12 bits		5 bits	5 bits
B-Type	opcode	immediate			
	6 bits	26 bits			
CB-Type	opcode	immediate		R_t	
	8 bits	19 bits		5 bits	

Setting Register Selects and ALUSrc

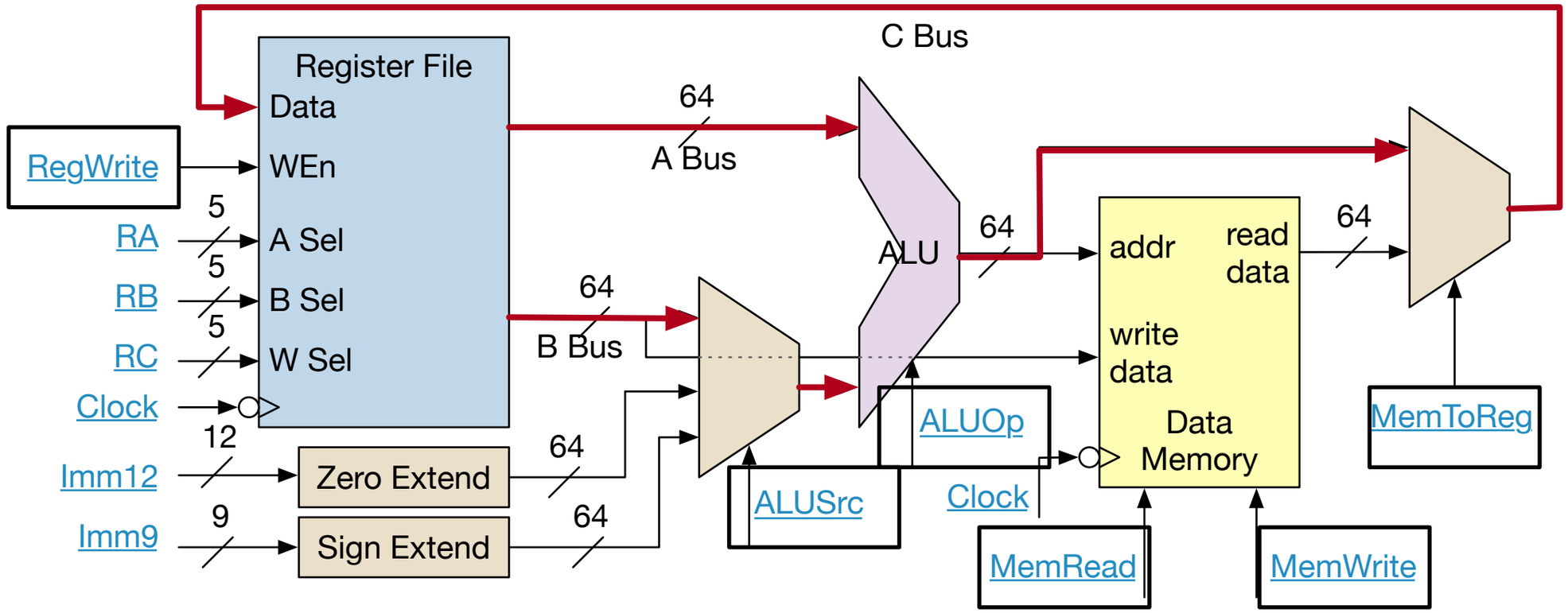
- Hypothetical implementation
  - Note use of splitter to extract only some bits



Single Cycle Datapath During **`add`** (shifted reg)

- Given RTN of **`X[d] ← X[m] + X[n]`**, then RegWrite = 1, ALUSrc = 0, ALUOp = add, MemRead = 0, MemWrite = 0, and MemToReg = 0

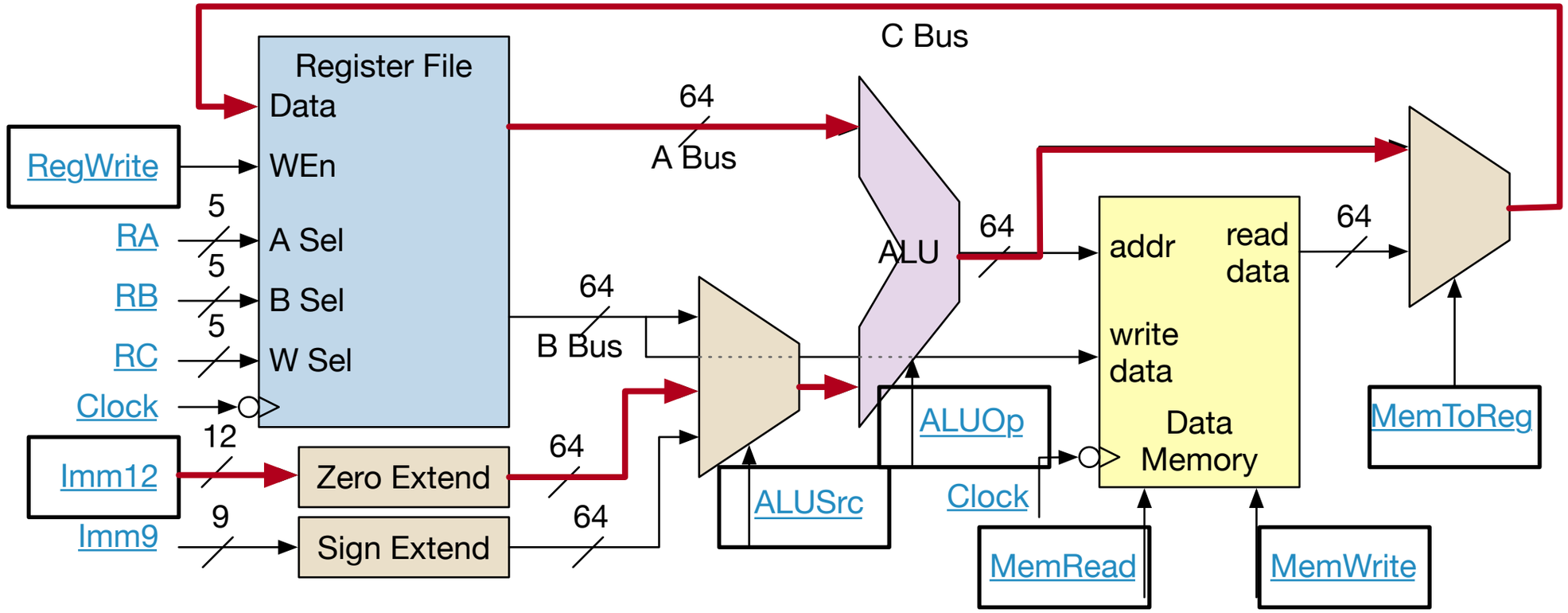
  - **`RA`** = **m**, **`RB`** = **n**, **`RC`** = **d**



Single Cycle Datapath During **`add`** (immediate)

- Given RTN of **`X[d] ← X[n] + ZeroExtend(imm12)`**, then RegWrite = 1, ALUSrc = 1, ALUOp = add, MemRead = 0, MemWrite = 0, and MemToReg = 0

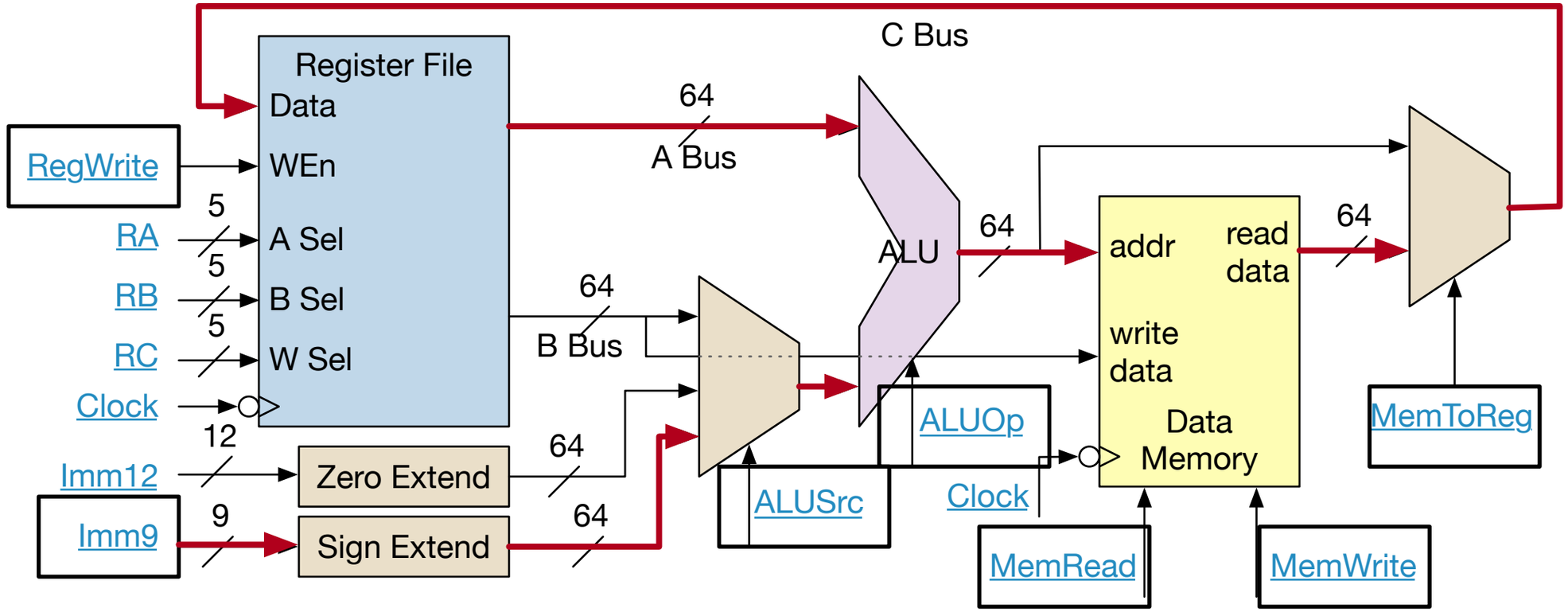
  - **`RB`** will be set to a don't care value; instead **`imm12`** will be set



Single Cycle Datapath During `ldur`

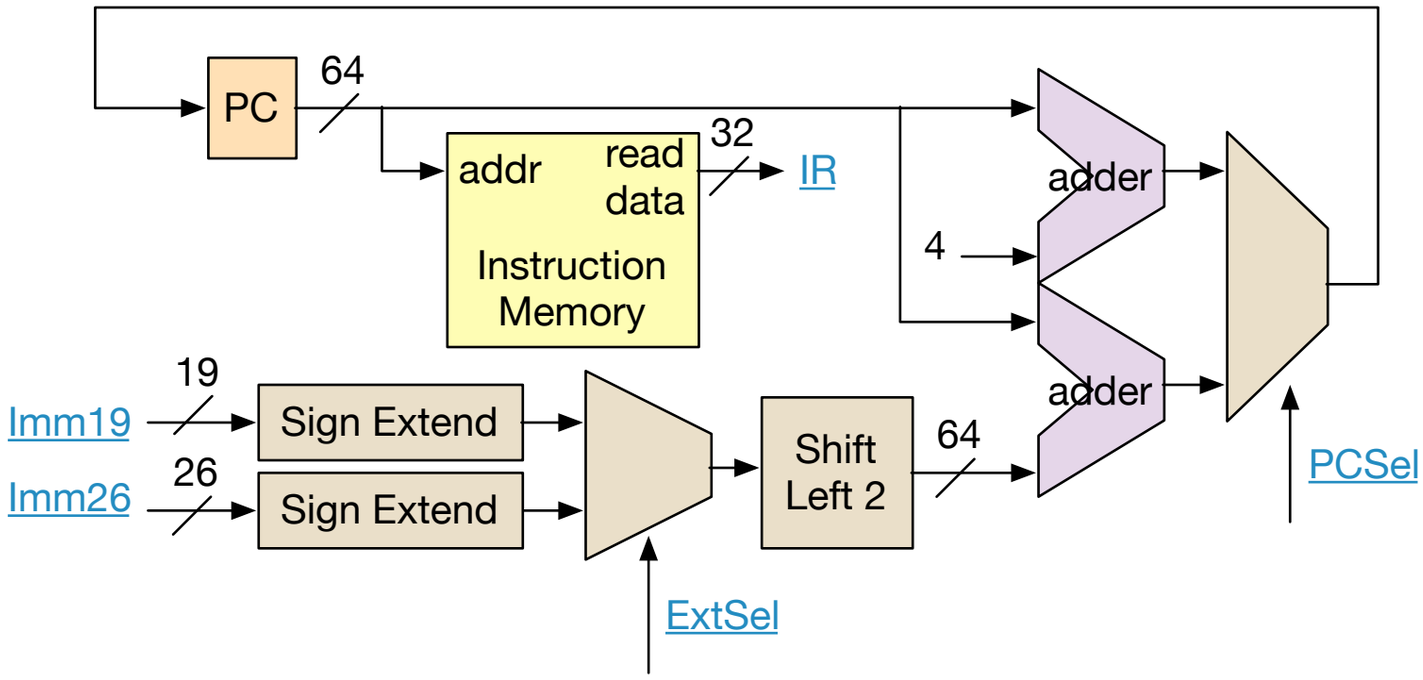
- Given RTN of `X[t] ← Mem[X[n] + SignExtend(imm9)]`, then RegWrite = 1, ALUSrc = 2, ALUOp = add, MemRead = 1, MemWrite = 0, and MemToReg = 1

  - Note how long this datapath is



Instruction Fetch Unit

- Instruction at **`PC`** is read into the instruction register and/or into decoder
  - **`PC`** is then normally increased by 4, and PCSel control line is set to 0
  - Instead for a branch, a different address for **`PC`** is computed and PCSel = 1



Selecting PC Value

- Normally, PCSel is set to 0 so that a mux selects **`PC`** ← **`PC`** + 4

- To jump to a different address, PCSel is set to 1:

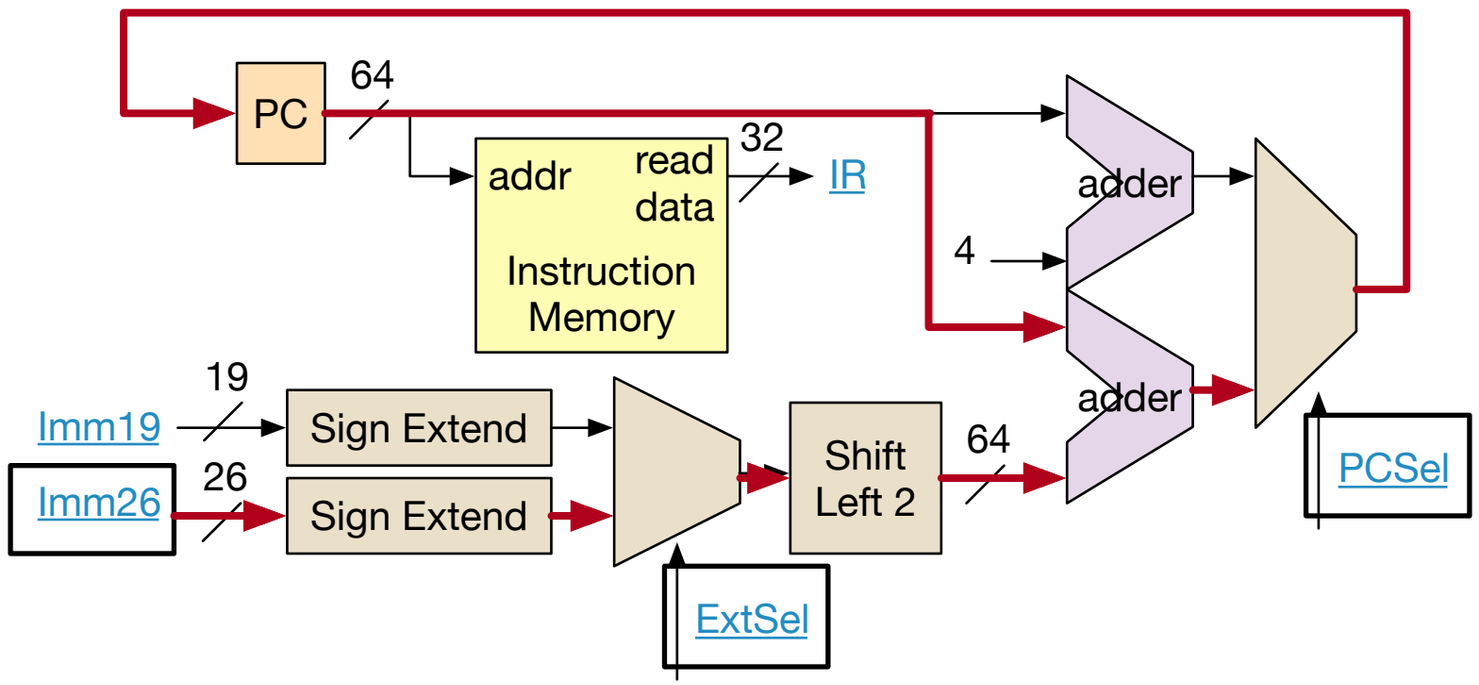
  - For an unconditional jump (such as **`b`**), ExtSel = 1; **`imm26`** is extracted from instruction

  - For a conditional jump if a register is / is not zero (**`cbz`** or **`cbnz`**), ExtSel = 0; **`imm19`** is extracted (and **`imm26`** is set to don't care)

  - For a conditional jump based upon ALU's condition codes (**`b.cond`**), ExtSel = 0, but PCSel is 1 only if condition is true

Single Cycle Datapath During `b`

- Given RTN of **`PC ← PC + (SignExtend(imm26)) × 4`**, then ExtSel = 1 and PCSel = 1
    - Shift left by 2 is equivalent to multiplying by 4, for unsigned integers
- Read **`PC`** on leading edge, write updated value on falling edge



Conditional Branching

- Some instructions set condition flags (Z, C, N, and V) as a side-effect of execution

- For **`b.cond`**, branch control unit sets PCSel = 1 if the following is true:

cond	Mnemonic	Meaning (Integer)	Condition Flags
0000	eq	Equal	Z == 1
0001	ne	Not equal	Z == 0
1010	ge	Signed greater than or equal	N == V
1011	lt	Signed less than	N != V
1100	gt	Signed greater than	Z == 0 && N == V
1101	le	Signed less than or equal	!(Z == 0 && N == V)

ARM Architecture Reference Manual for A-profile architecture, Section C1.2.4

Summary of Control Signals

- Could implement entire instruction set as a giant Karnaugh map

Control	Instruction		
	R-Type	ldur	b.*cond*
RegWrite	1	1	0
ALUSrc	00	10	XX
ALUOp	based upon opcode	add	X
MemRead	0	1	X
MemWrite	0	0	0
MemToReg	0	1	X
ExtSel	X	X	1
PCSel	0	0	based upon branch control unit

Worst Case Timing (Load)

- Clock cycle must be greater than longest path (which is often a memory load)