CSE 31
Computer Organization

Lecture 23 – CPU Design (2)

Announcement

- No lab this week
  - Project #1 grading during lab
- HW #7 at zyBooks
  - Due Monday(12/10) at 11:59pm
- Project #2
  - Due Monday (12/3)
    - Don’t start late, you won’t have time!
- Course evaluation online
  - Fill out by 12/6 (Thursday)
- Reading assignment
  - Chapter 5.7 – 5.11 of zyBooks (Reading Assignment #6)
    - Make sure to do the Participation Activities
    - Due Wednesday (12/5)

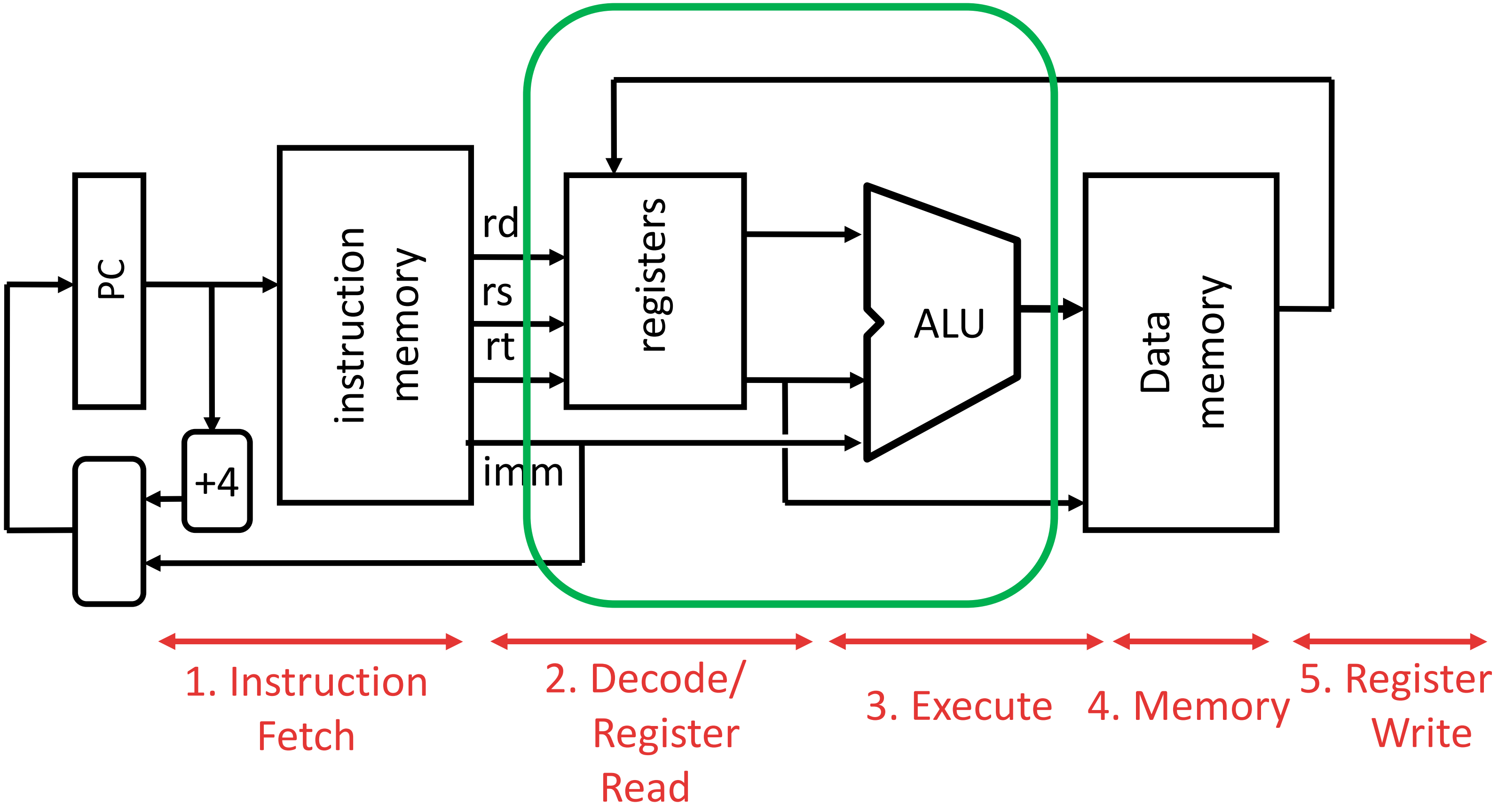
Announcement

- Final Exam
  - 12/11 (Tuesday), 11:30 – 2:30pm
  - Cover all
  - Practice exam in CatCourses
  - Closed book
  - 2 sheet of note (8.5” x 11”)
  - MIPS reference sheet will be provided
  - Review: 12/10 (Monday) 1-3pm, COB 113

Review

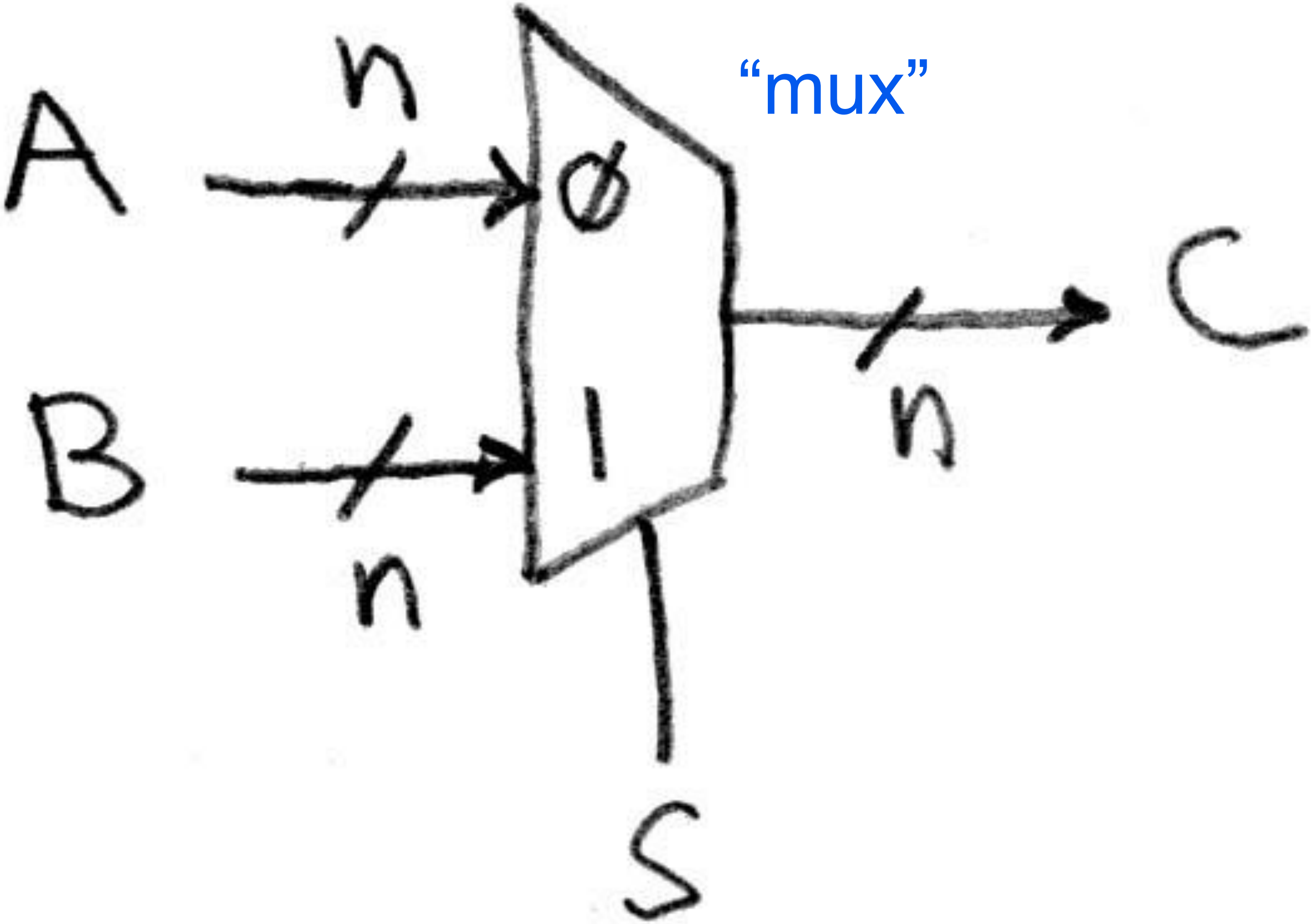
- CPU design involves Datapath,Control
  - Datapath in MIPS involves 5 CPU stages
    1. Instruction Fetch
    2. Instruction Decode & Register Read
    3. ALU (Execute)
    4. Memory
    5. Register Write

Generic Steps of Datapath



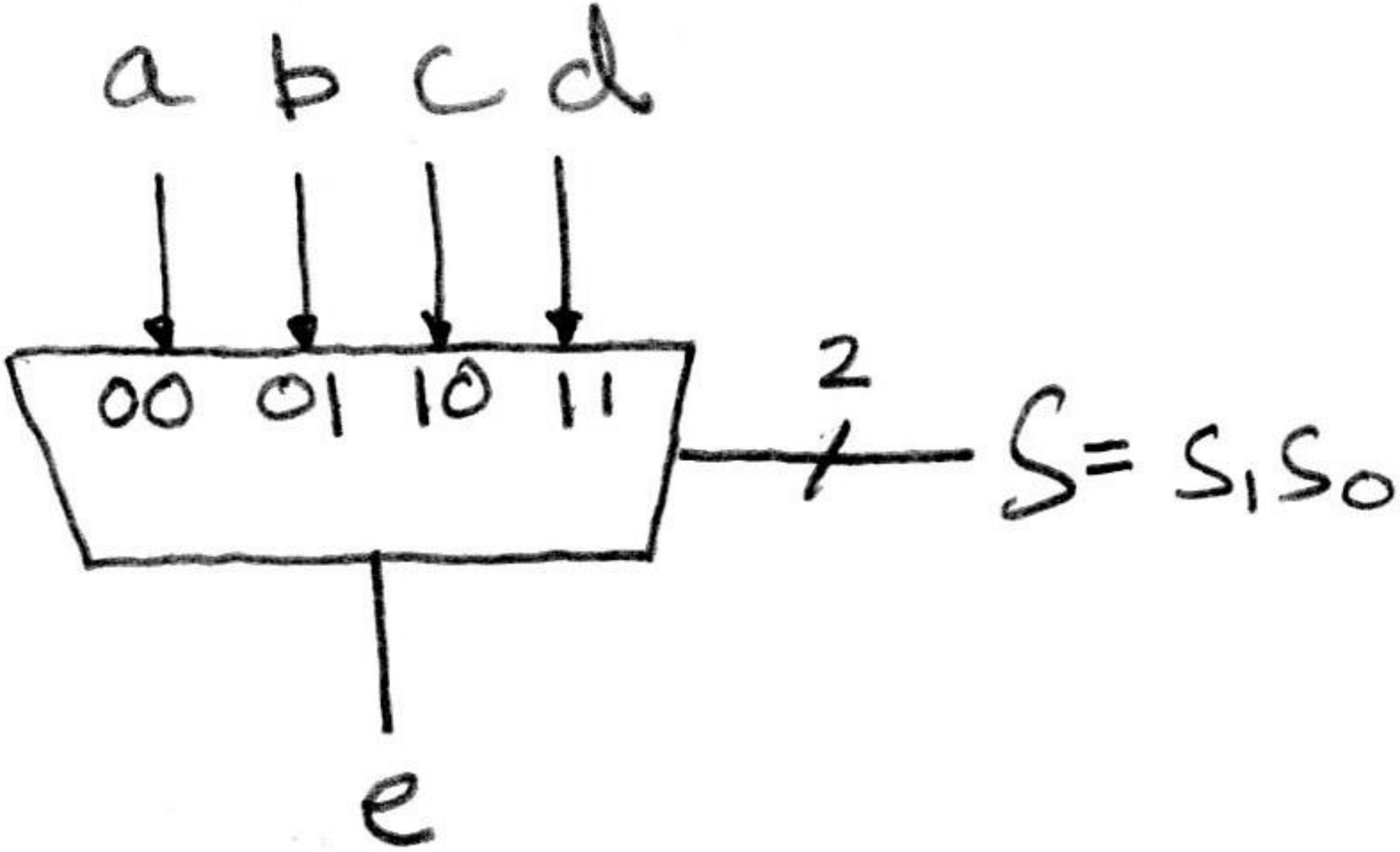
How do we handle the different register usage between r-type and i-type instructions?

Data Multiplexor (mux) (2-to-1, n-bits)

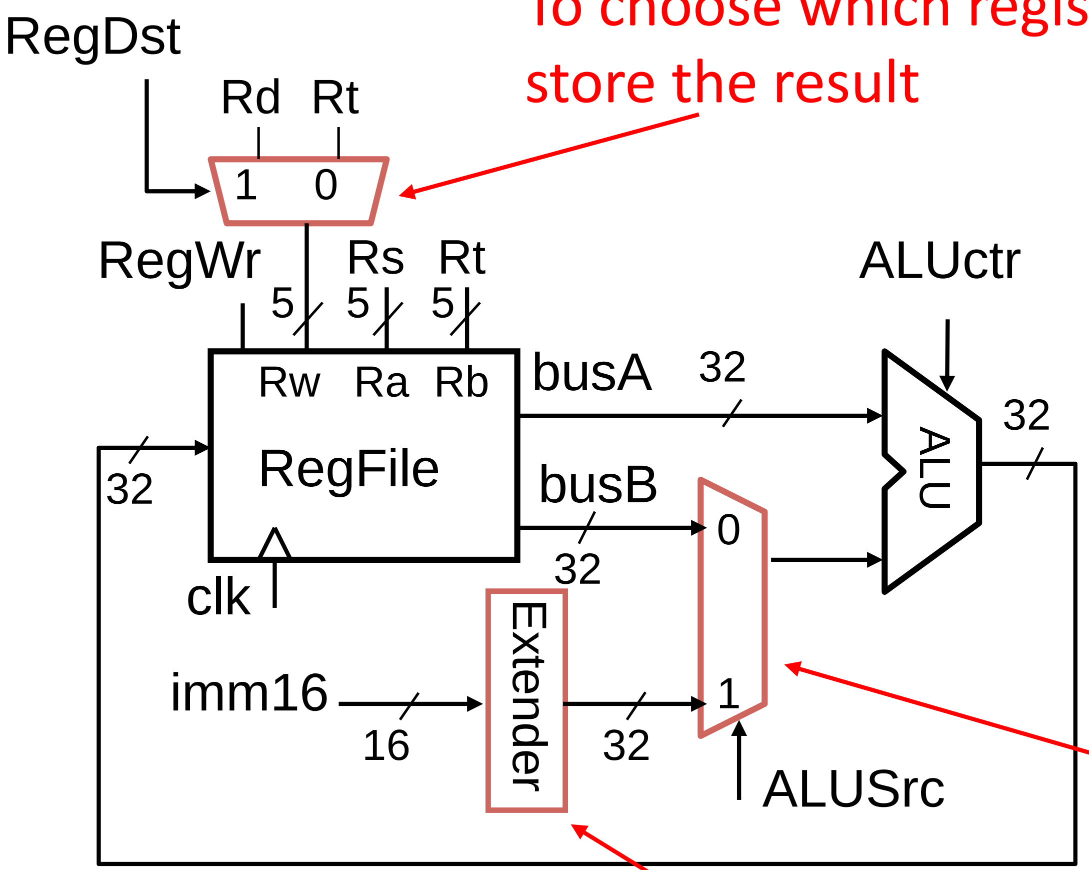


"mux"

4-to-1 Multiplexor?



A zoomed in version of RegFile and ALU



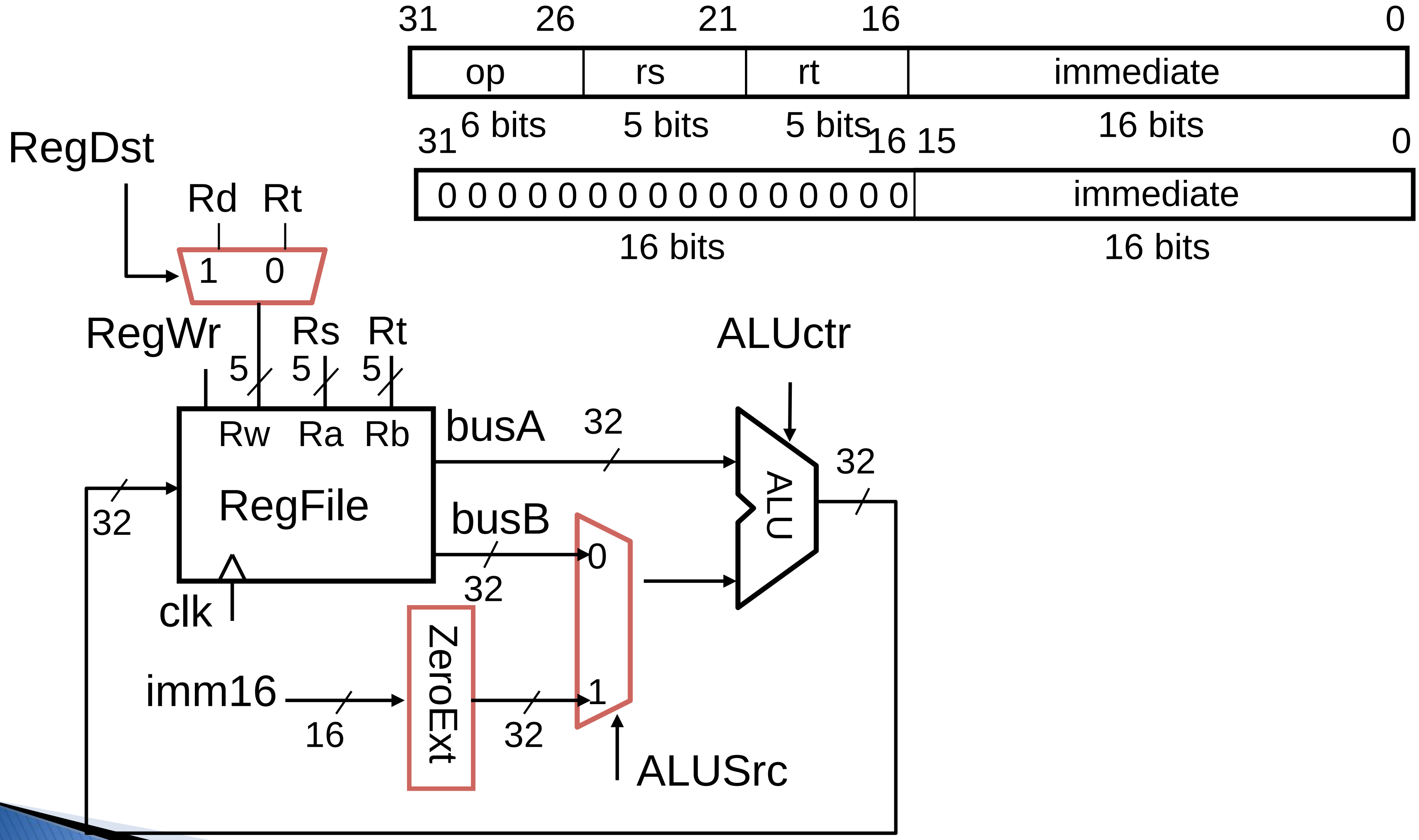
To choose which register to store the result

To choose which input to ALU

To extend 16 bits into 32 bits

Operations with Immediate

- R[rt] = R[rs] op ZeroExt[imm16]]



31 26 21 16 0

op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

31 16 15 0

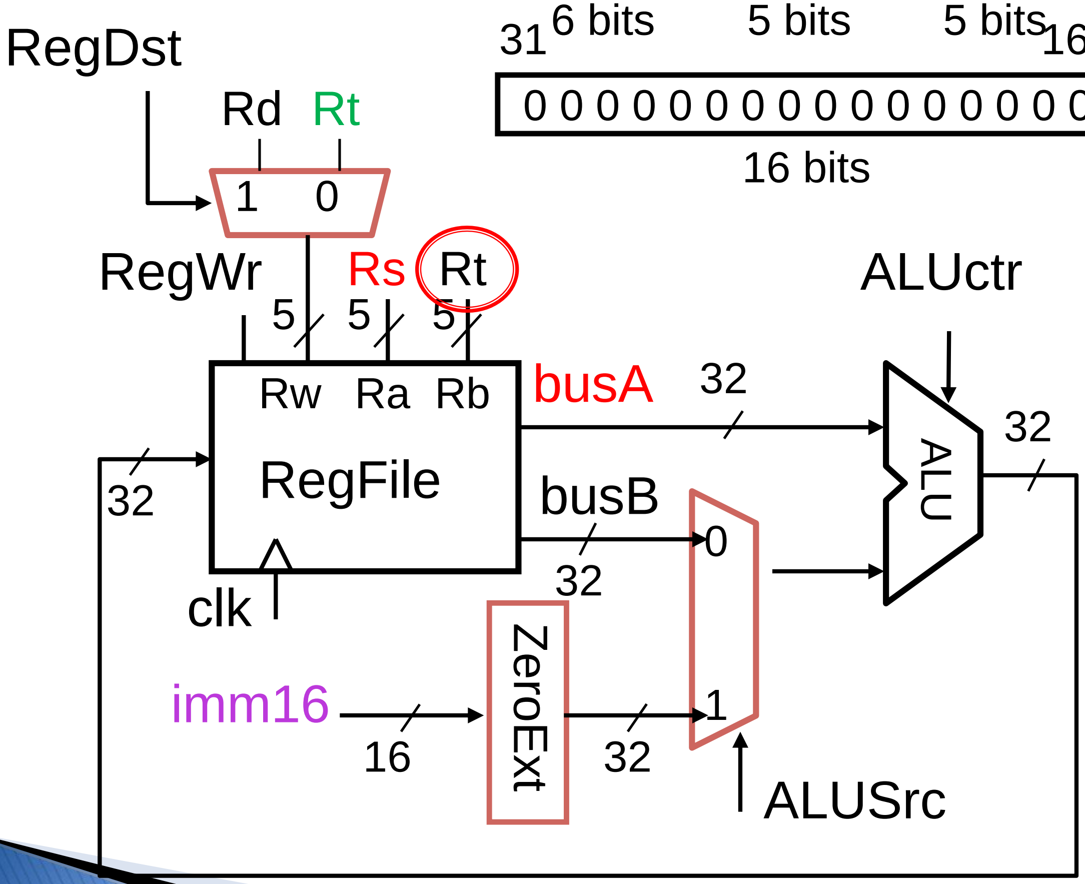
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	immediate
16 bits	16 bits

Operations with Immediate

- R[rt] = R[rs] op ZeroExt[imm16]]

31	26	21	16	0
op	rs	rt	immediate	
6 bits	5 bits	5 bits	16 bits	

31	16 15	0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	immediate	
16 bits	16 bits	



What about Rt register read (bus B)??

We don’t care!

Why?

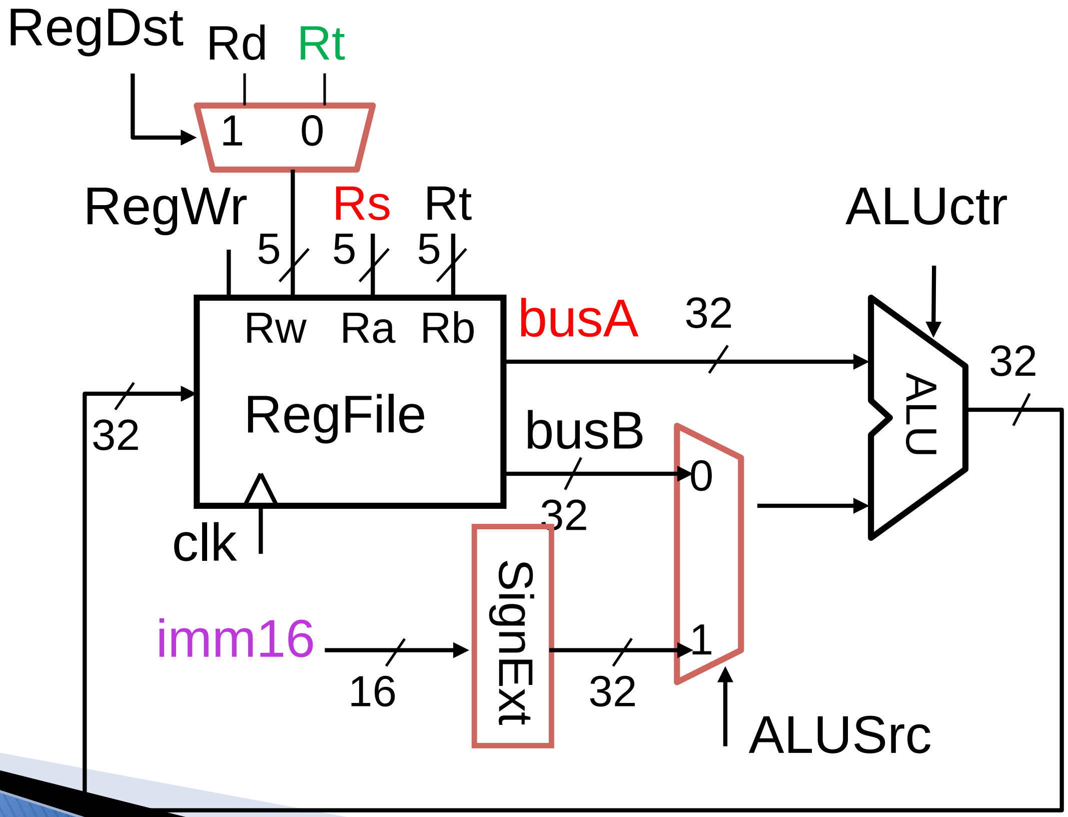
Bus B is blocked

Load Memory

- R[rt] = Mem[R[rs] + SignExt[imm16]]
- Example: `lw rt,rs,imm16`

31 26 21 16 0

op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits



Do we store the result of ALU?

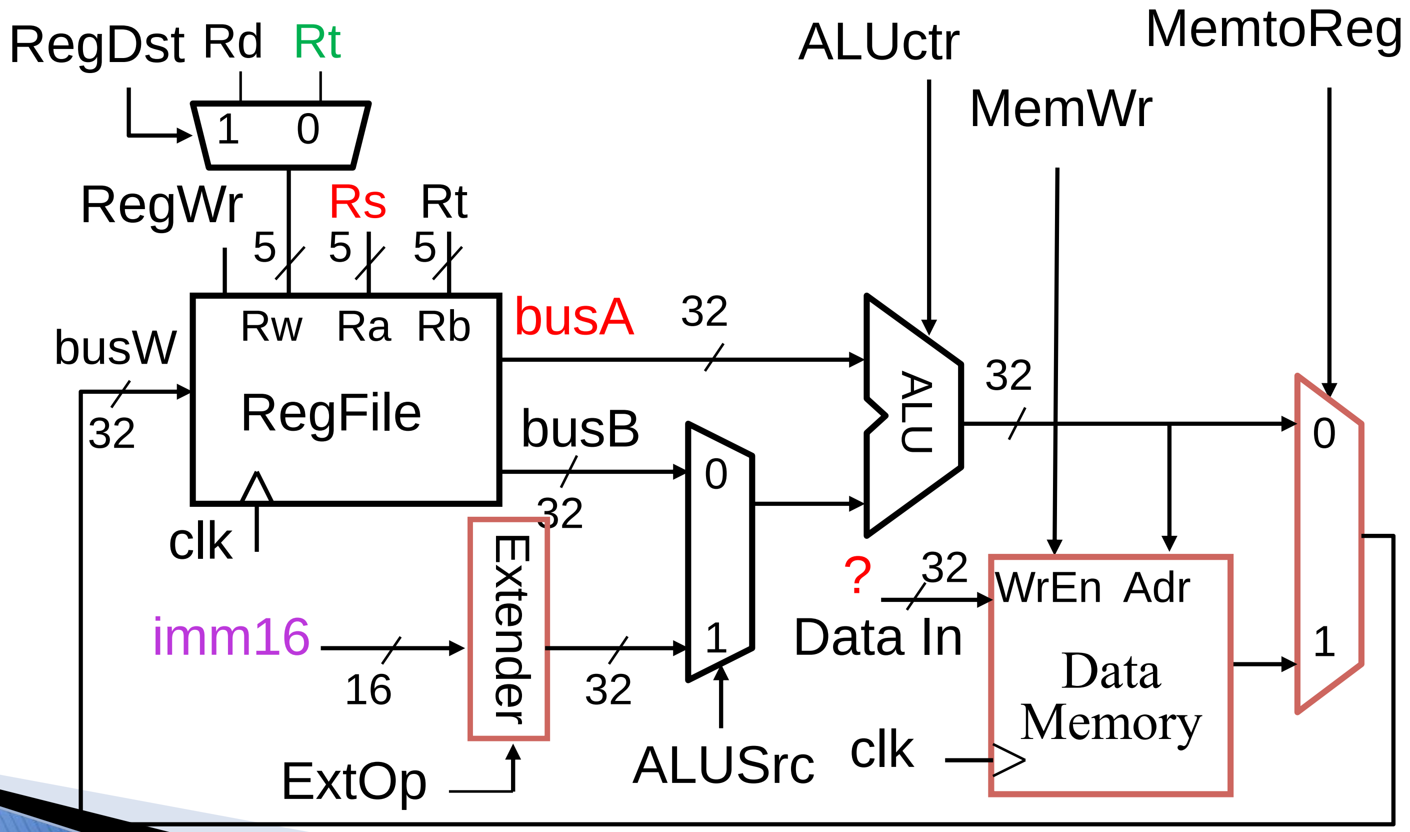
What is the result of ALU for?

We need to add a memory unit!

Load Memory

- R[rt] = Mem[R[rs] + SignExt[imm16]]
- Example: `lw rt,rs,imm16`

31	26	21	16	0
op	rs	rt	immediate	
6 bits	5 bits	5 bits	16 bits	

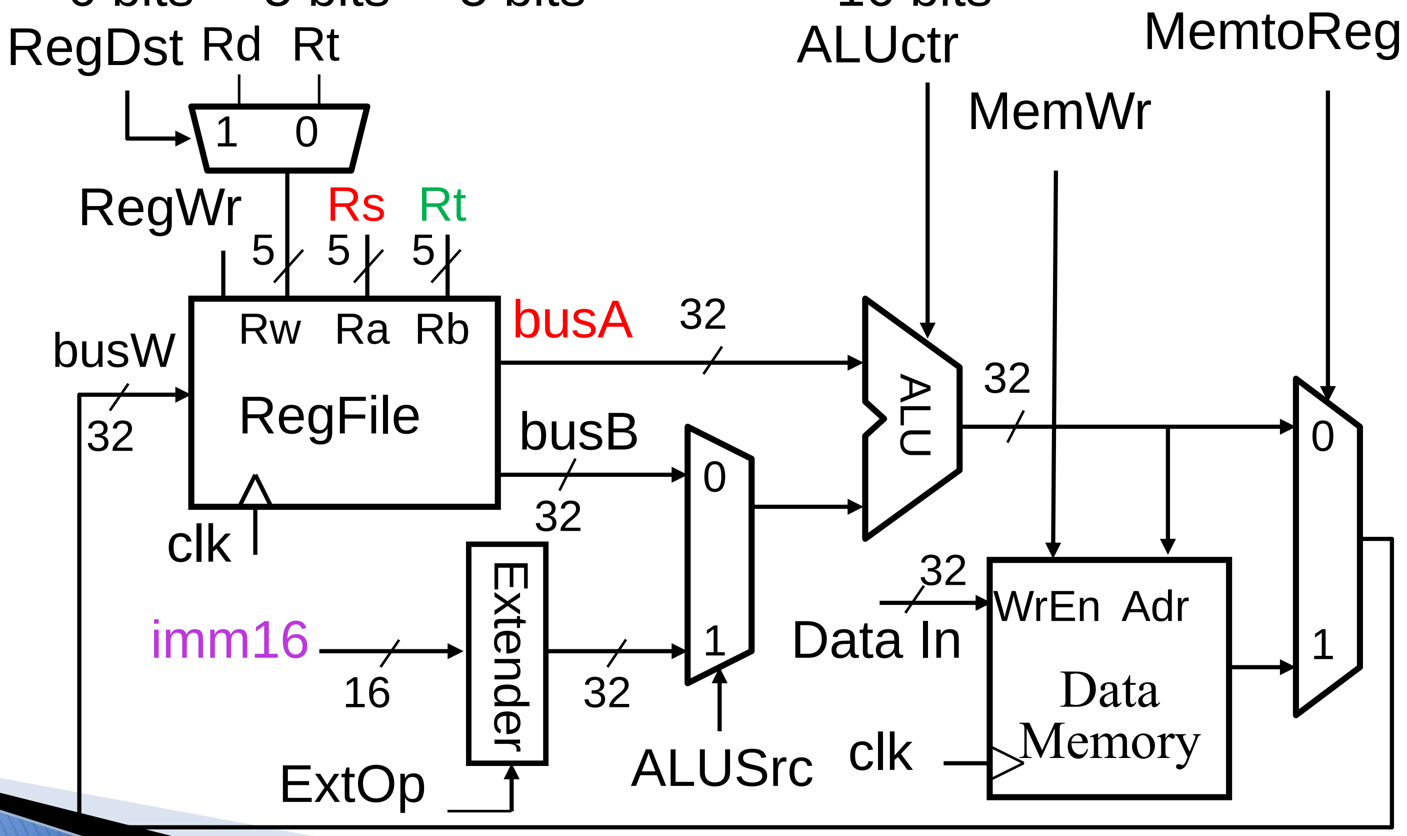


Store Memory

- Mem[R[rs] + SignExt[imm16]] = R[rt]

Ex.: `sw rt, rs, imm16`

31	26	21	16	0
op	rs	rt	immediate	
6 bits	5 bits	5 bits	16 bits	



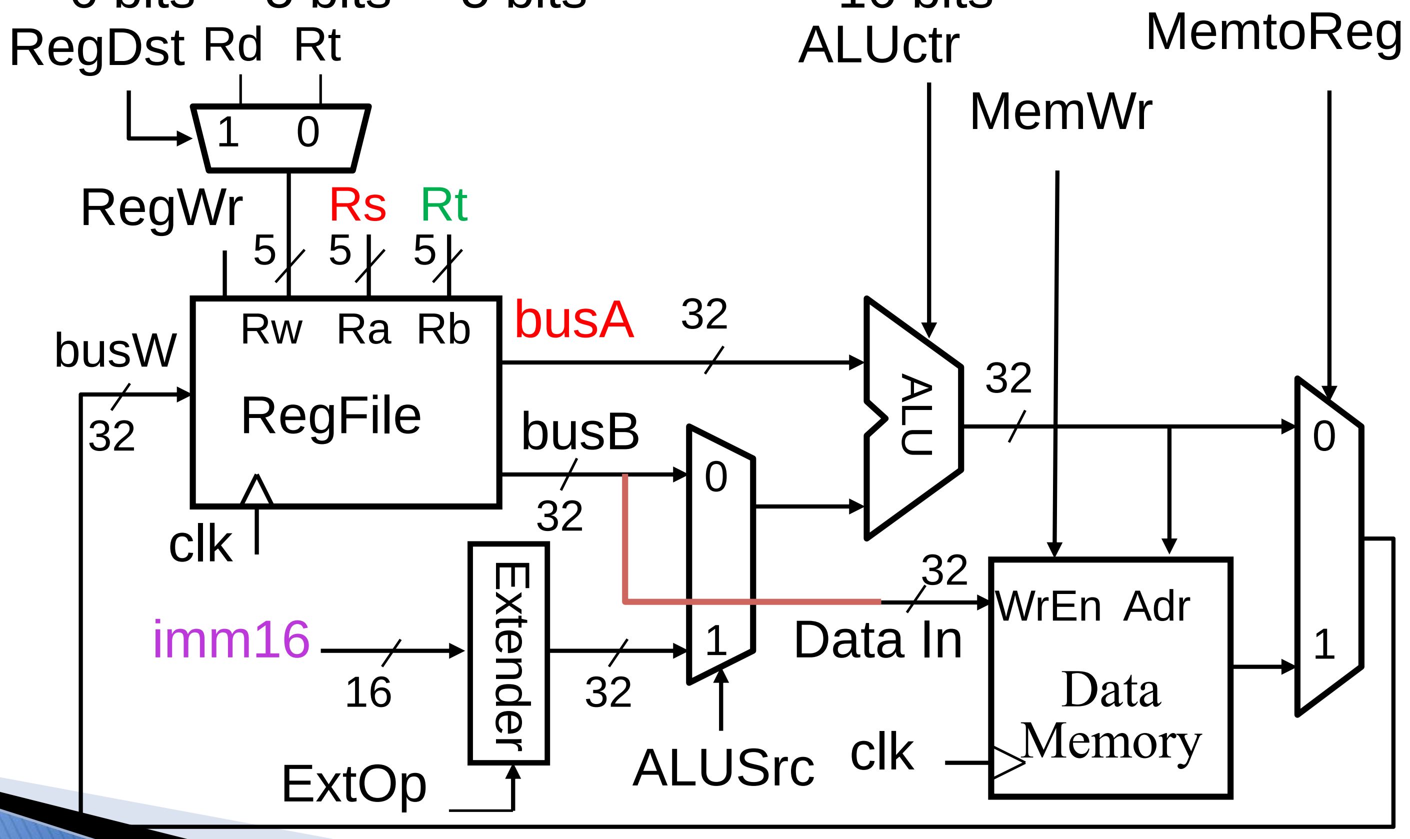
Store Memory

- Mem[R[rs] + SignExt[imm16]] = R[rt]

  Ex.: `sw rt, rs, imm16`

31 26 21 16 0

op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits



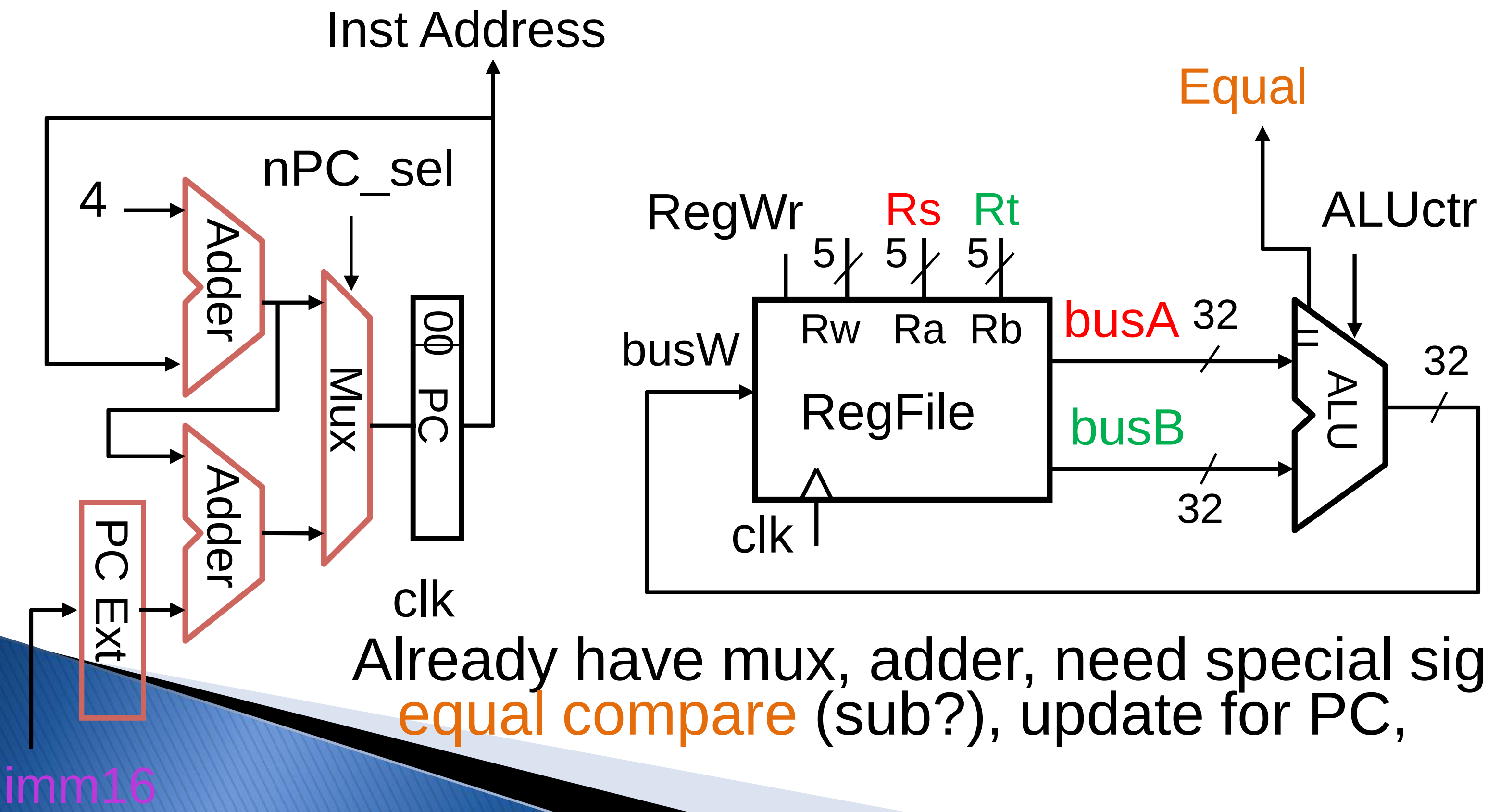
Datapath for Branch Operations

- beq rs, rt, imm16

  Datapath generates condition (equal)

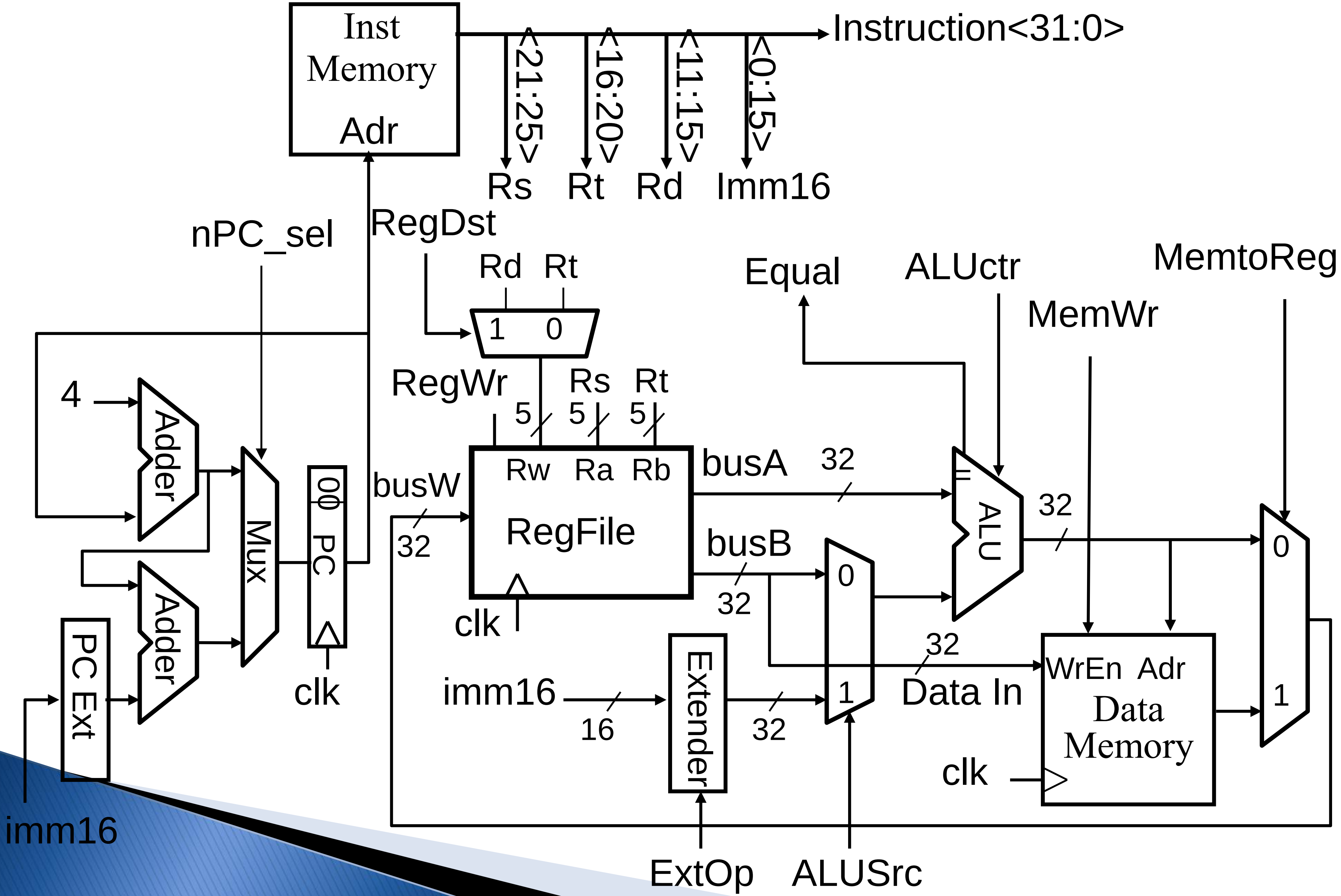
31 26 21 16 0

op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits

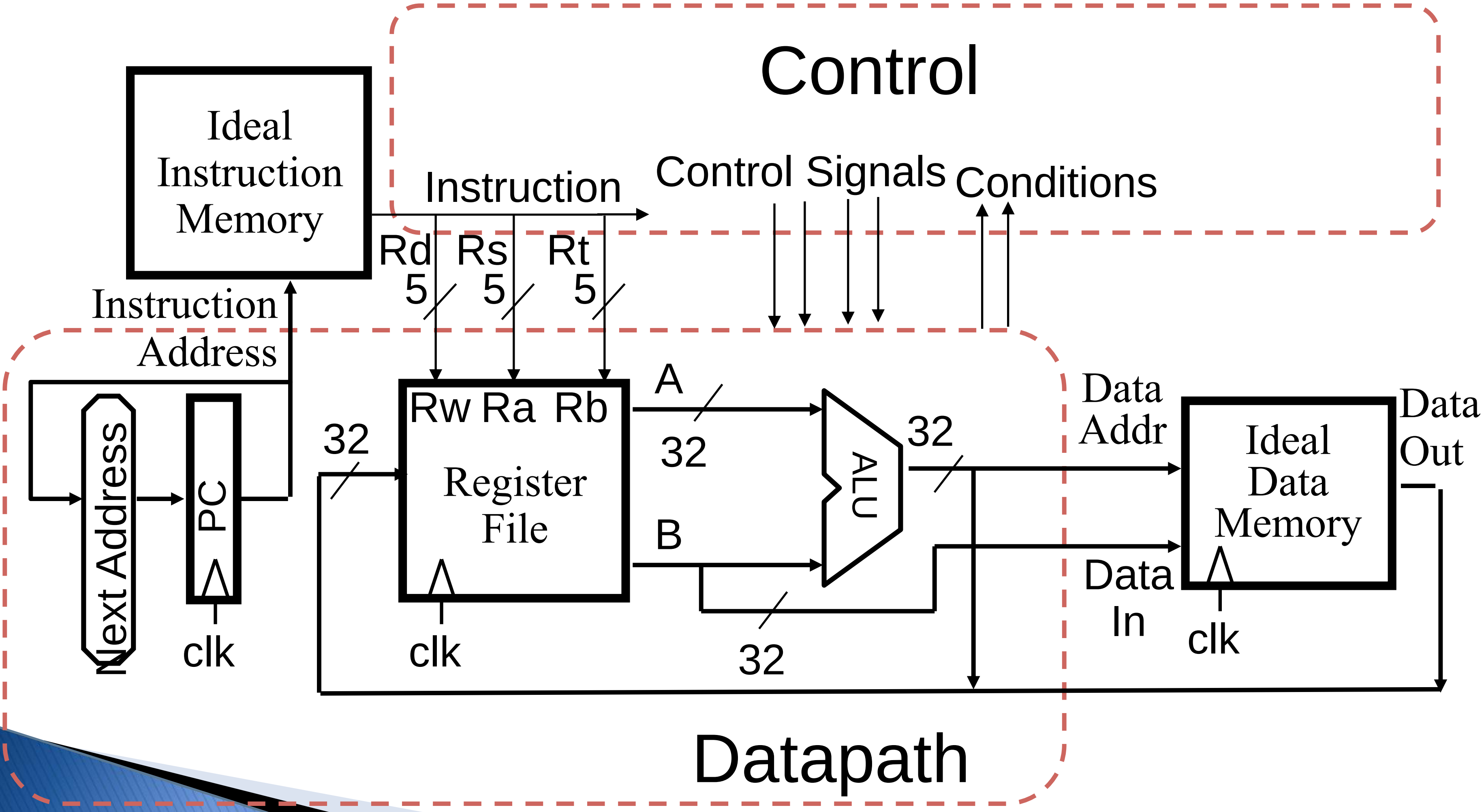


Already have mux, adder, need special sign need equal compare (sub?), update for PC,

Single Cycle Datapath

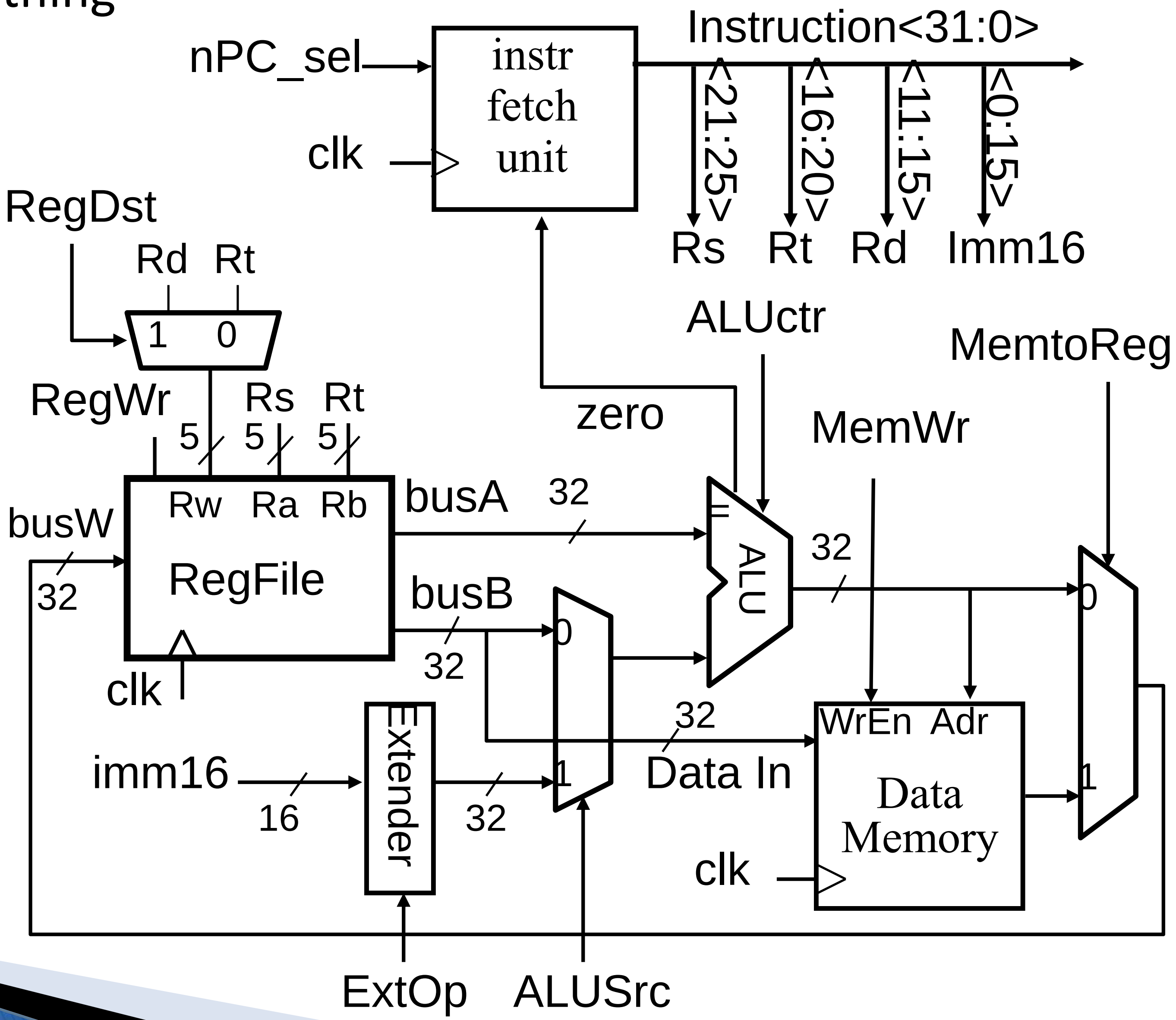


Abstract View of the Implementation



A Single Cycle Datapath

- We have everything except control signals

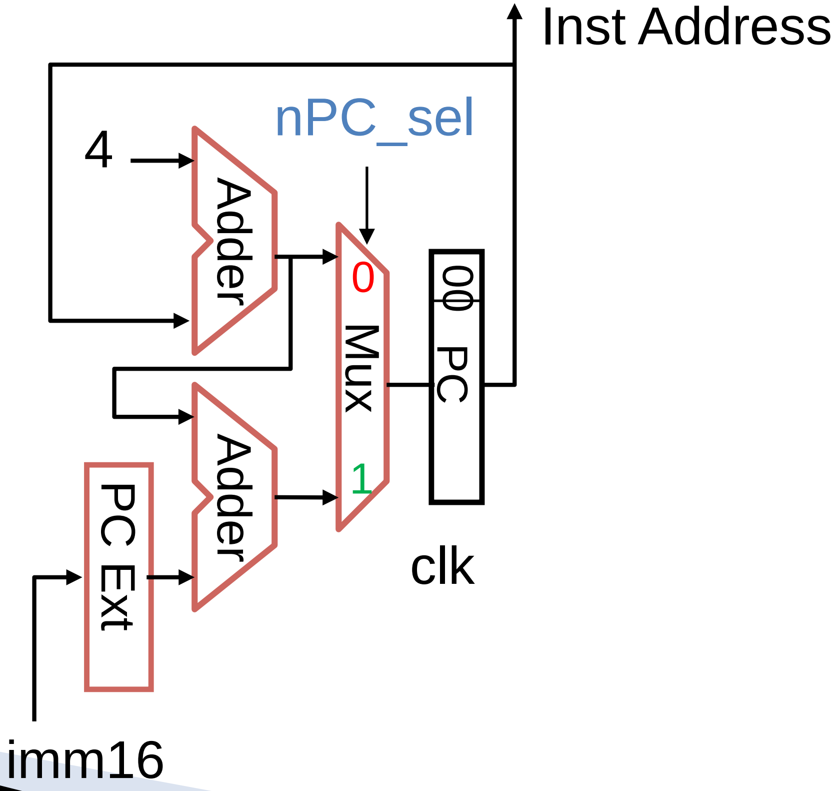


Meaning of the Control Signals

- nPC_sel: “+4”: 0 ⇒ PC <– PC + 4
  “br”: 1 ⇒ PC <– PC + 4 + {SignExt(Im16) , 00 }

  “n”=next

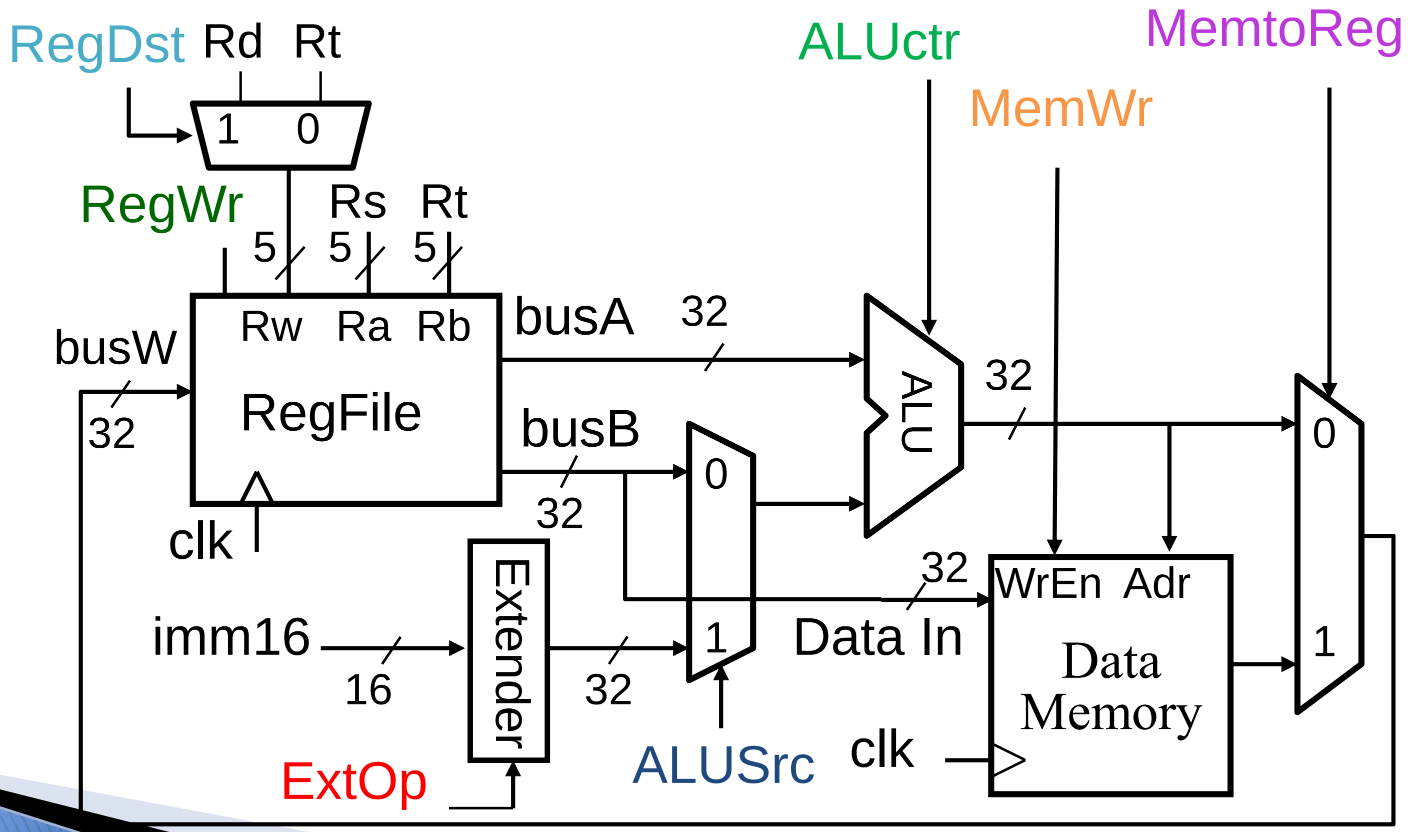
- Later in lecture: higher-level connection between mux and branch condition



Meaning of the Control Signals

- ExtOp: “zero”, “sign”
- ALUsrc: 0 ⇒ regB;
  1 ⇒ immed
- ALUctr: “ADD”, “SUB”, “OR”

- MemWr: 1 ⇒ write memory
- MemtoReg: 0 ⇒ ALU; 1 ⇒ Mem
- RegDst: 0 ⇒ “rt”; 1 ⇒ “rd”
- RegWr: 1 ⇒ write register



The Add Instruction

31 26 21 16 11 6 0

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

`add rd, rs, rt`

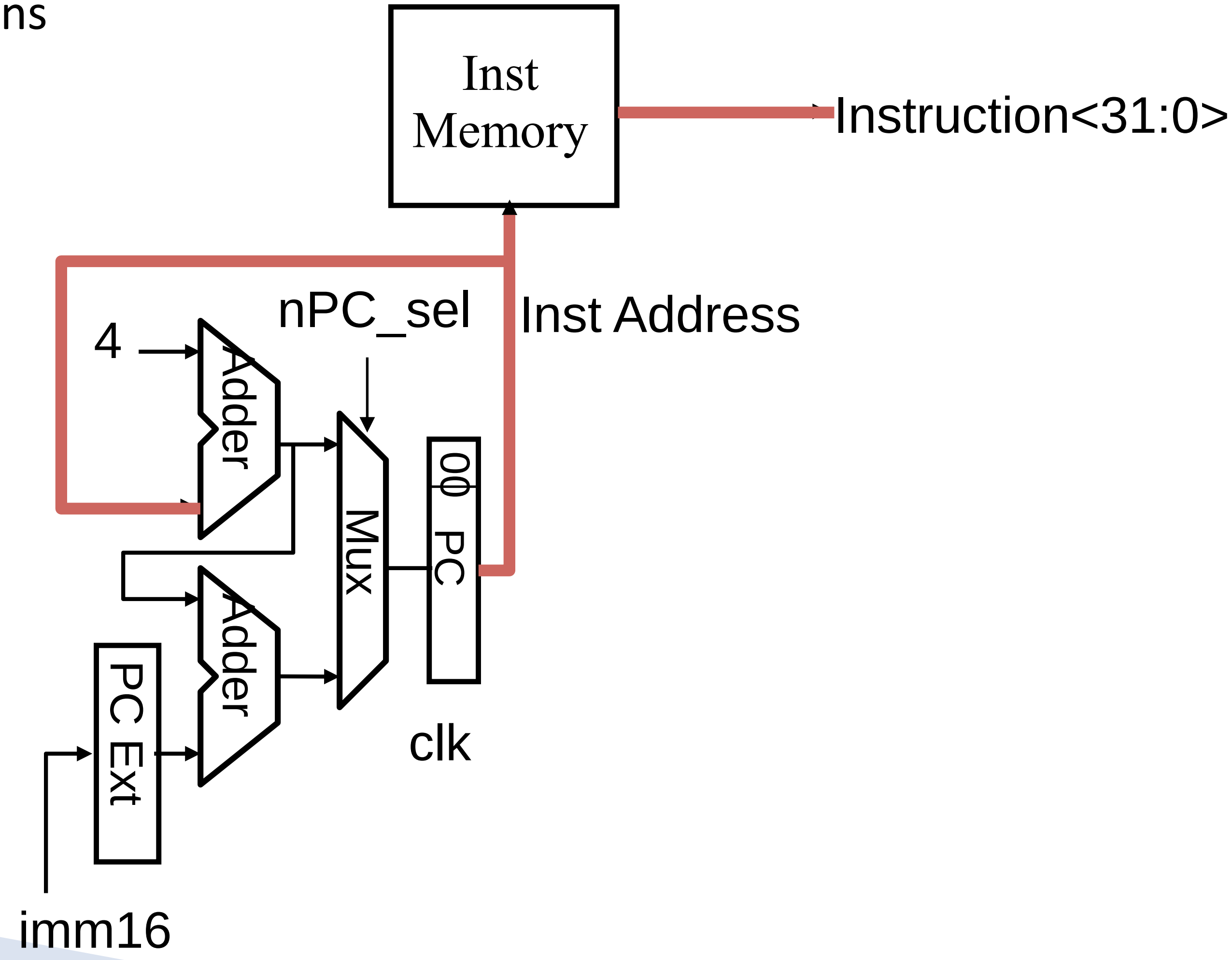
- MEM[PC] Fetch the instruction from memory
- R[rd] = R[rs] + R[rt] The actual operation
- PC = PC + 4 Calculate the next instruction’s address

Instruction Fetch Unit start of Add

- Fetch the instruction from Instruction memory:

  Instruction = MEM[PC]

  - same for all instructions

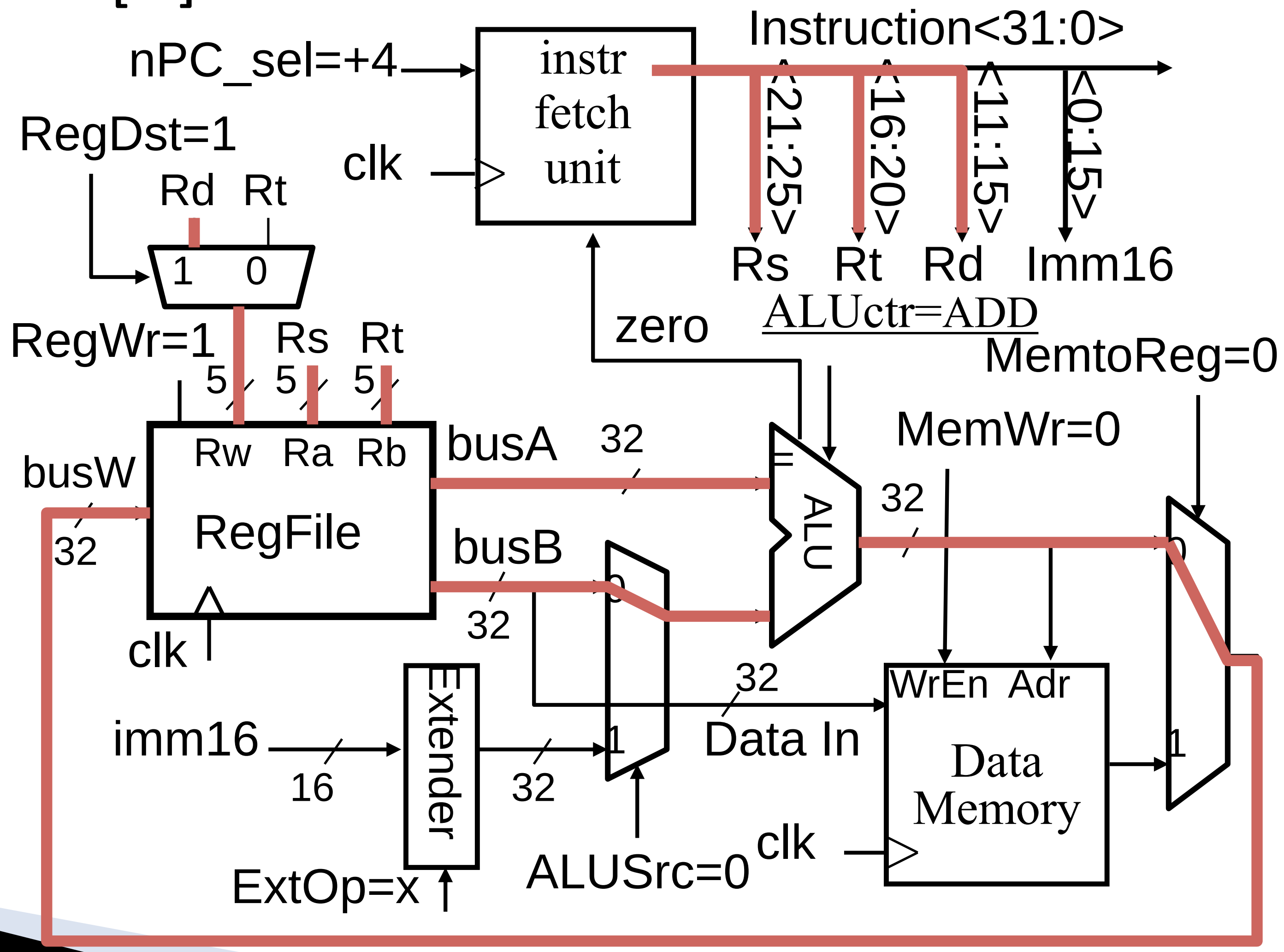


The Single Cycle Datapath during Add

31 26 21 16 11 6 0

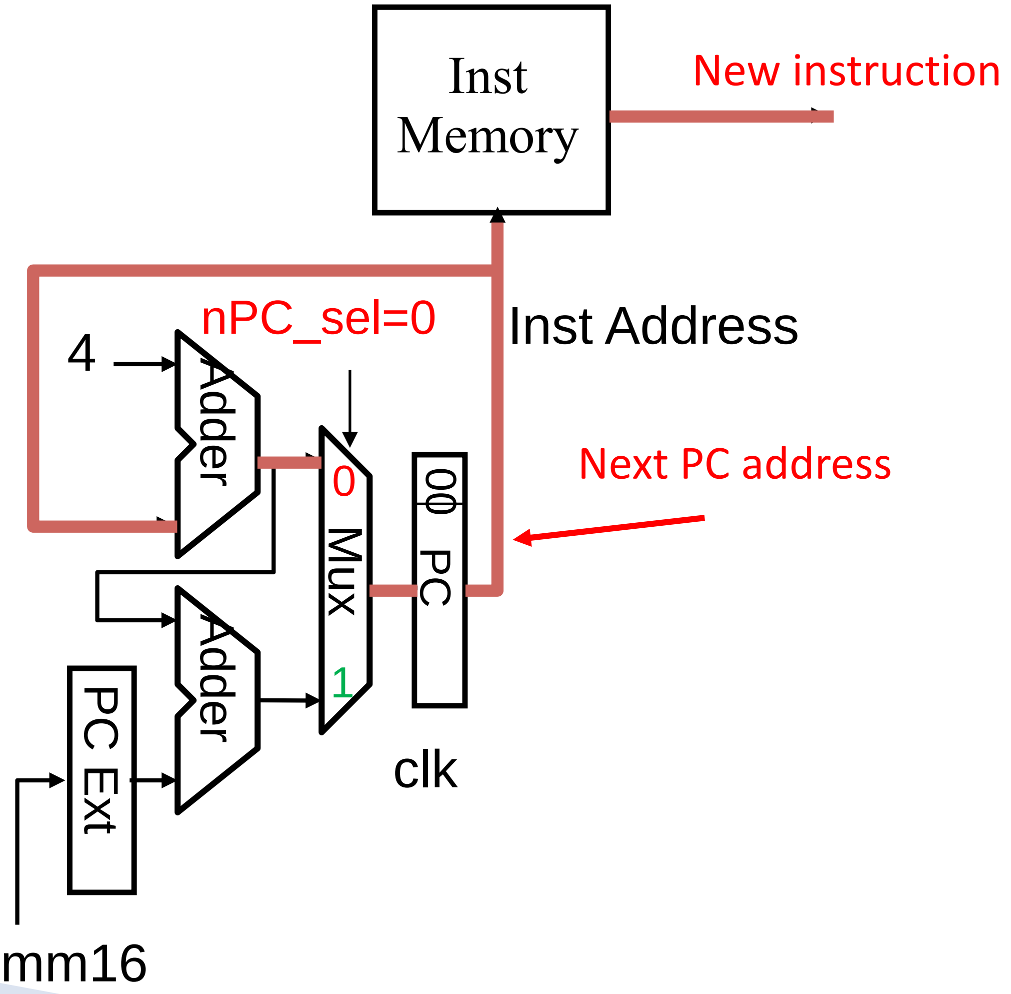
op	rs	rt	rd	shamt	funct

R[rd] = R[rs] + R[rt]



Instruction Fetch Unit end of Add

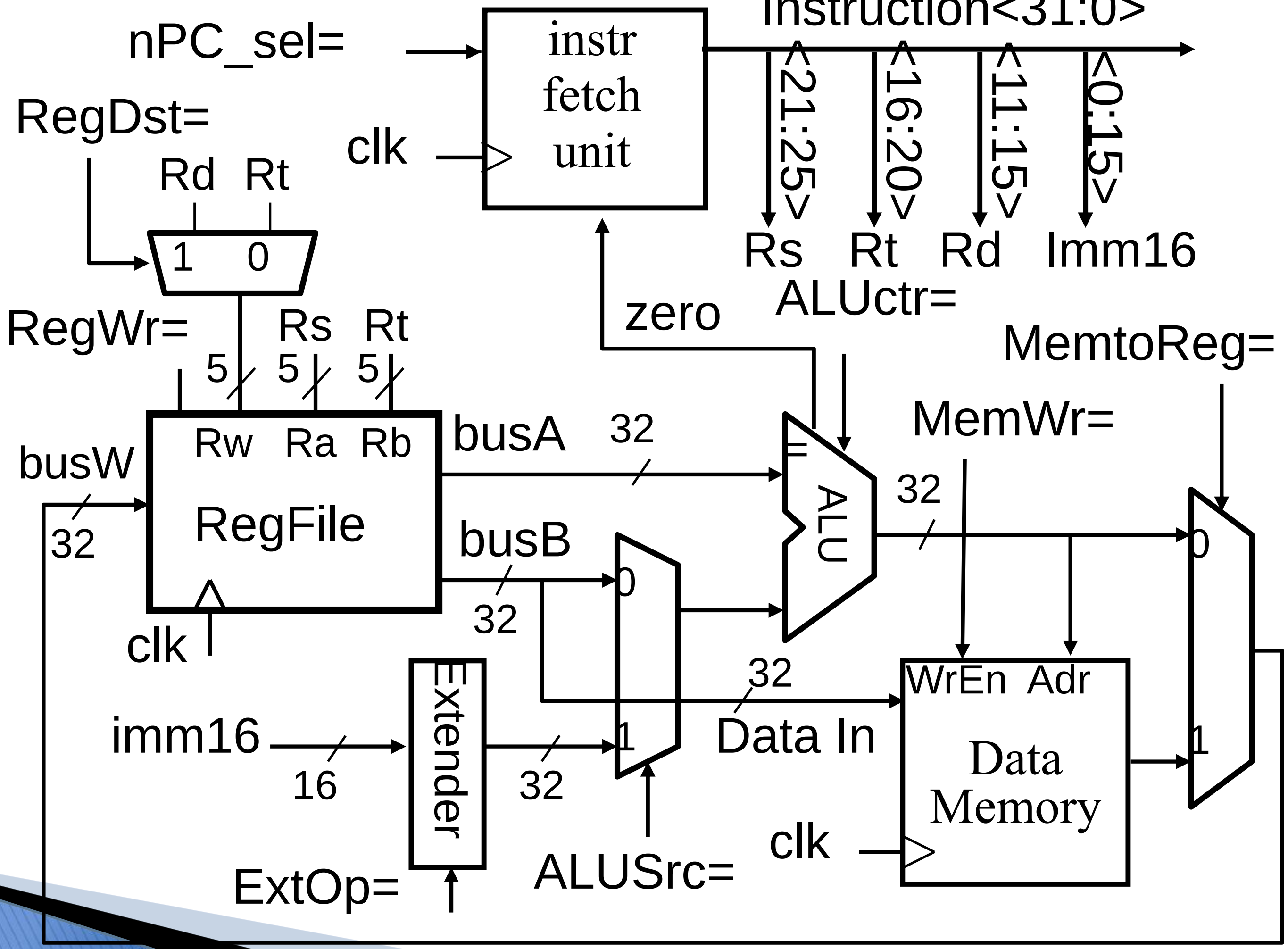
- PC = PC + 4
  - This is the same for all instructions except: Branch and Jump



Single Cycle Datapath for Ori

31	26	21	16	0
op	rs	rt	immediate	

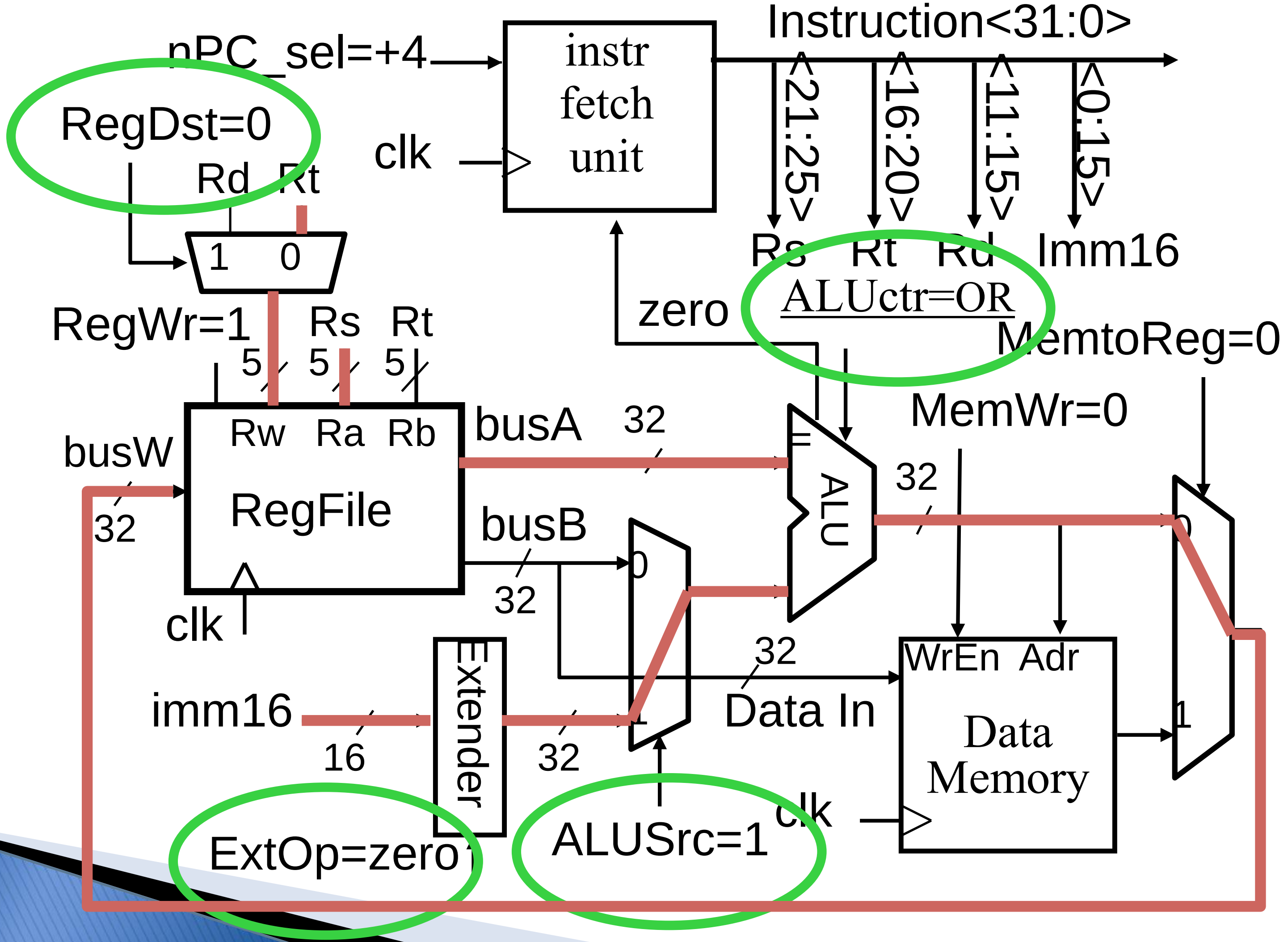
- R[rt] = R[rs] OR ZeroExt[Imm16]



Single Cycle Datapath for Ori

31	26	21	16 0
op	rs	rt	immediate

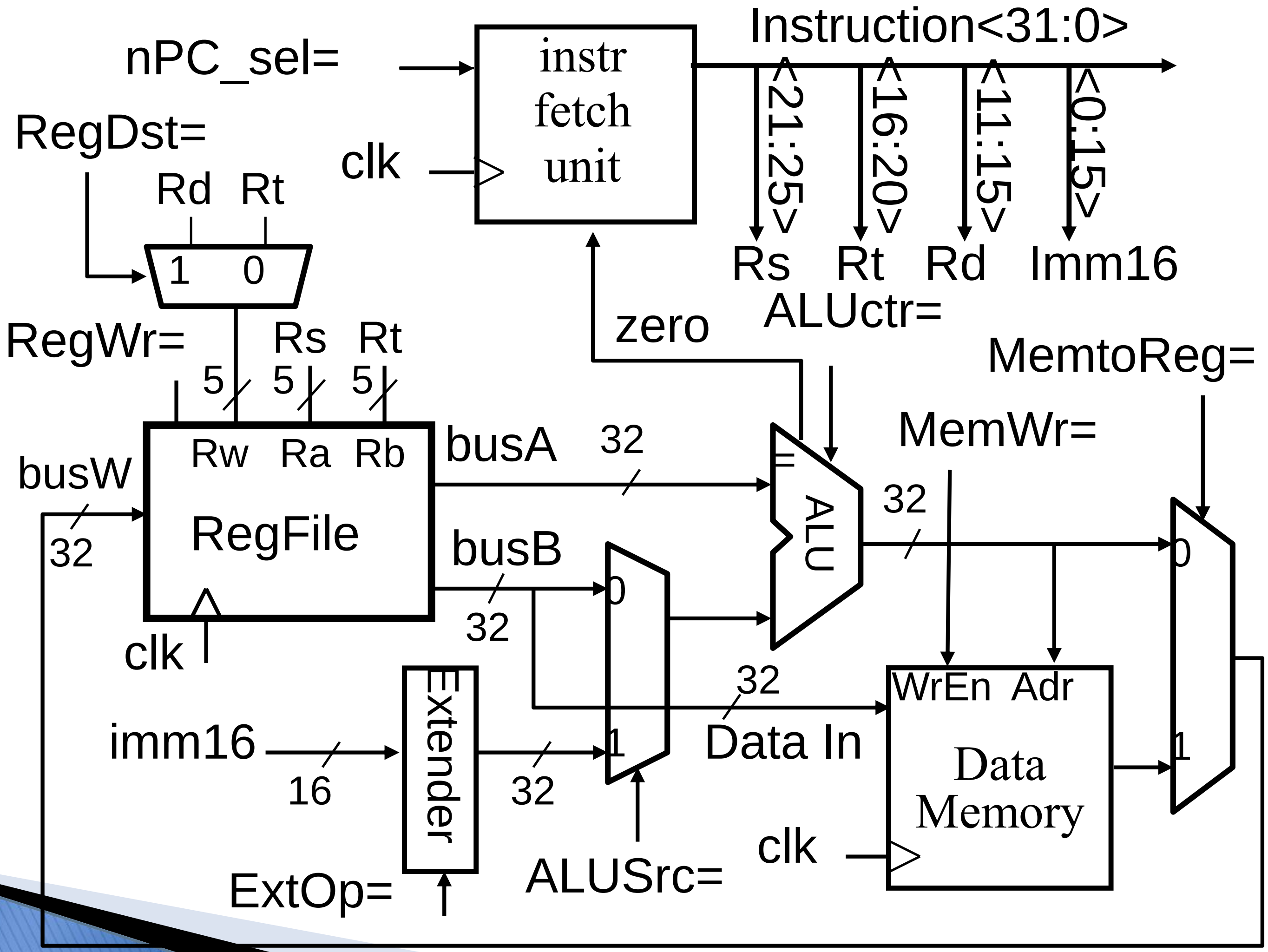
- R[rt] = R[rs] OR ZeroExt[Imm16]



Single Cycle Datapath for LW

31	26	21	16	0
op	rs	rt	immediate	

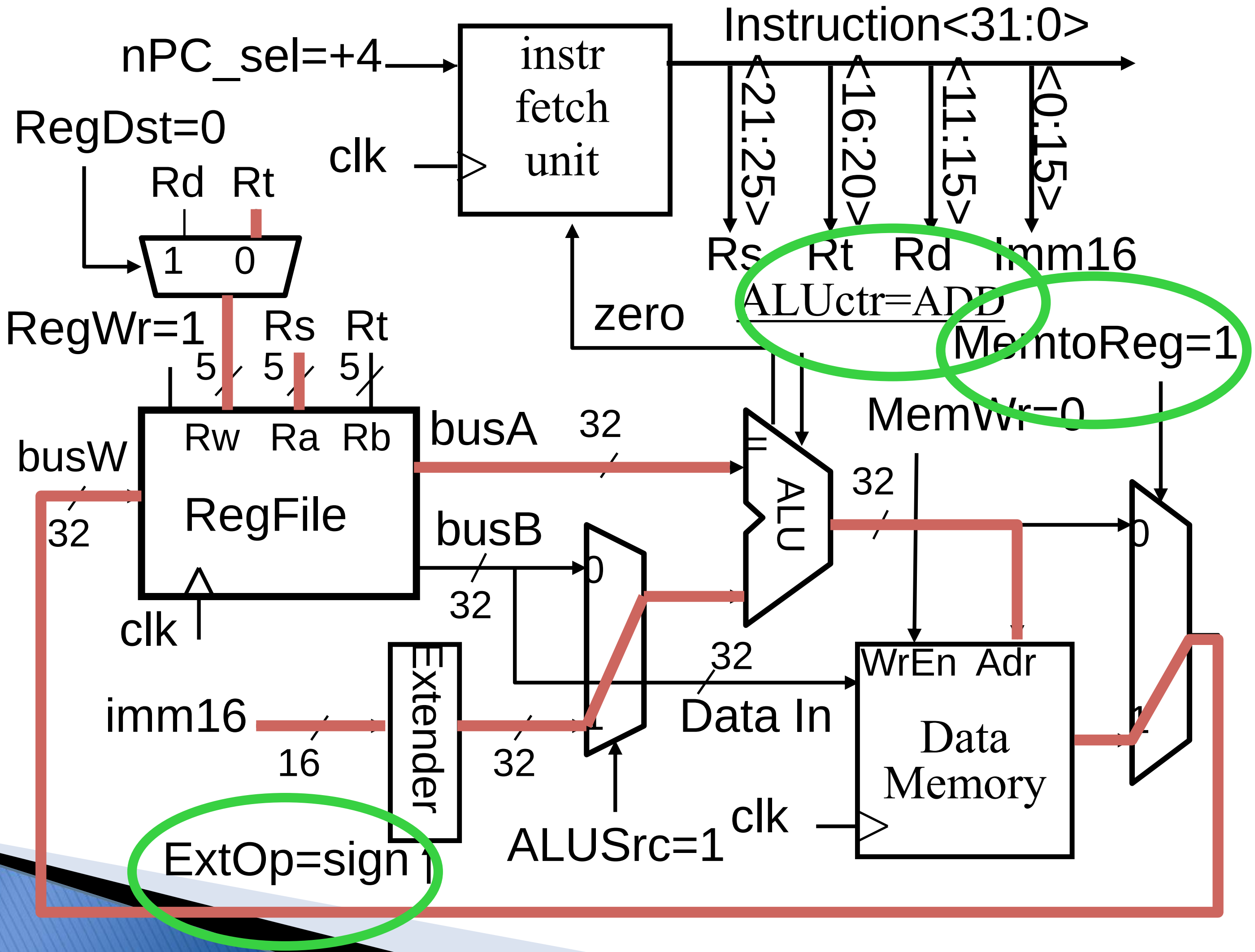
- R[rt] = Data Memory {R[rs] + SignExt[imm16]}



Single Cycle Datapath for LW

31	26	21	16	0
op	rs	rt	immediate	

- R[rt] = Data Memory {R[rs] + SignExt[imm16]}

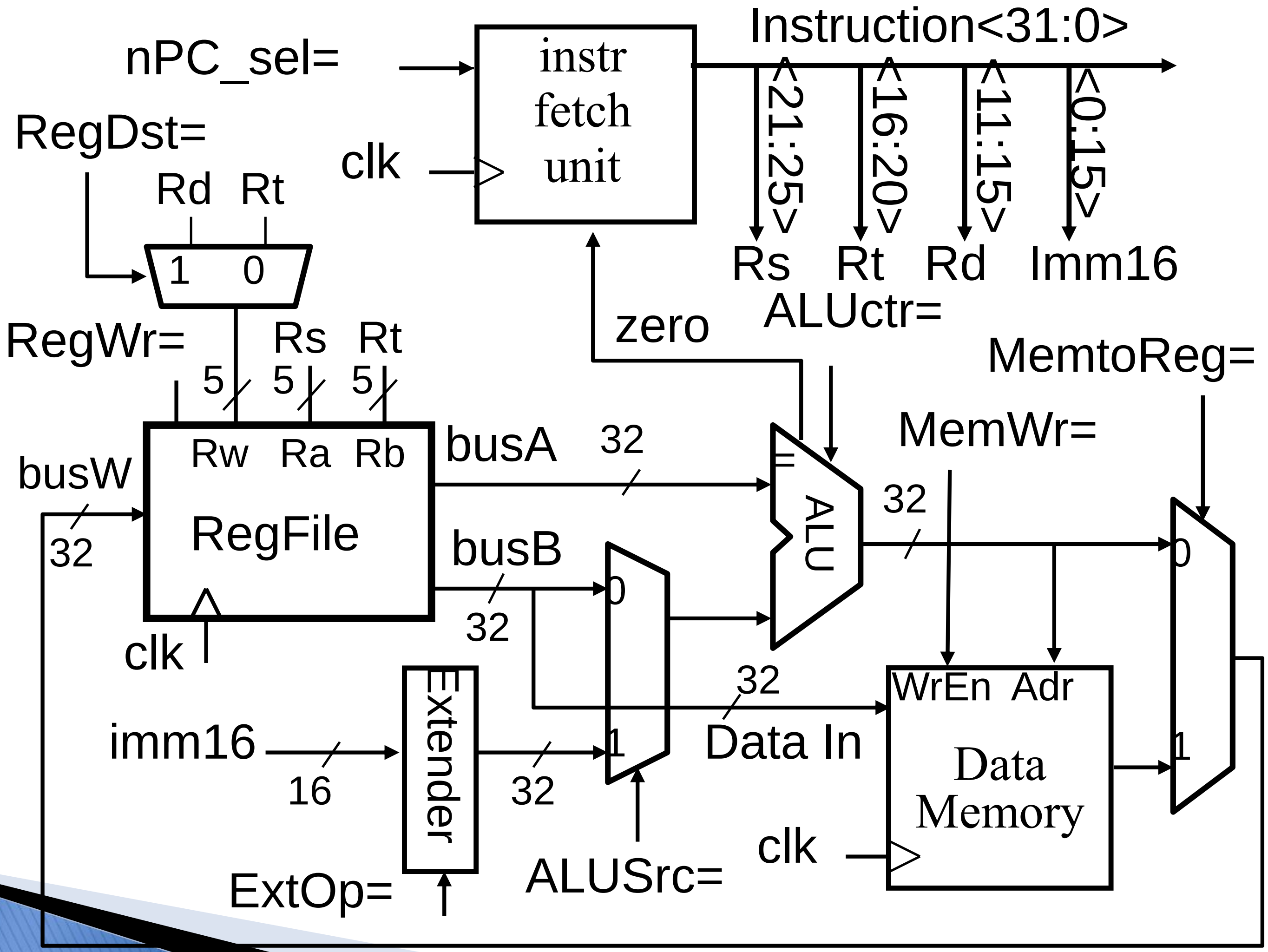


Single Cycle Datapath for SW

31 26 21 16 0

op	rs	rt	immediate

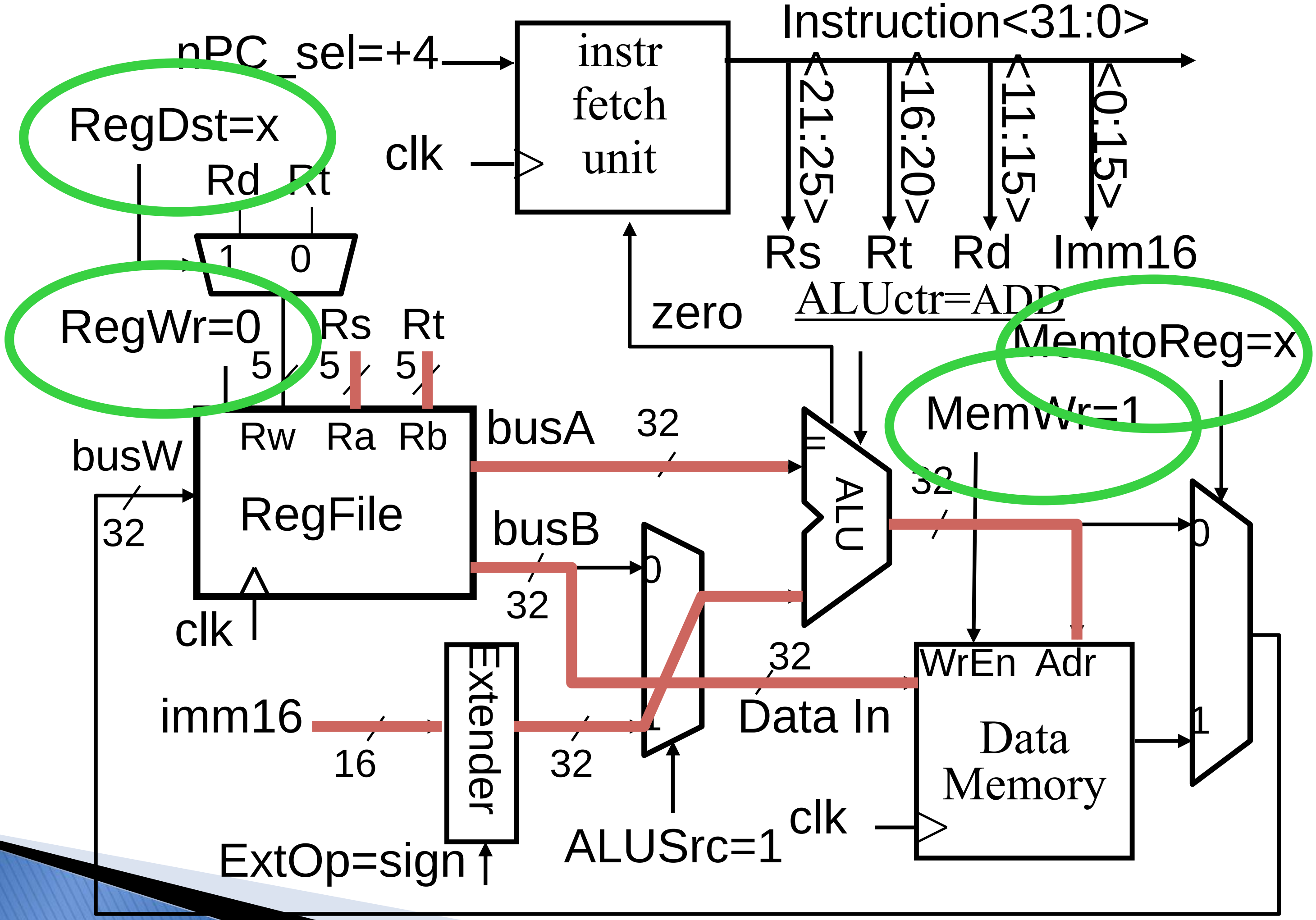
- Data Memory {R[rs] + SignExt[imm16]} = R[rt]



Single Cycle Datapath for SW

31	26	21	16	0
op	rs	rt	immediate	

- Data Memory {R[rs] + SignExt[imm16]} = R[rt]



nPC_sel=+4
instr fetch unit
clk
Instruction<31:0>
<21:25>
<16:20>
<11:15>
<0:15>
Rs
Rt
Rd
Imm16
RegDst=x
Rd
Rt
1
0
RegWr=0
Rs
Rt
5
5
5
zero
ALUctr=ADD
MemtoReg=x
MemWr=1
busW
32
Rw
Ra
Rb
RegFile
busA
32
busB
32
clk
ALU
32
0
1
0
1
imm16
16
Extender
32
32
Data In
WrEn
Adr
Data Memory
clk
ALUSrc=1
ExtOp=sign