CSE 31
Computer Organization

Lecture 25 – CPU Design (cont.)

Announcements

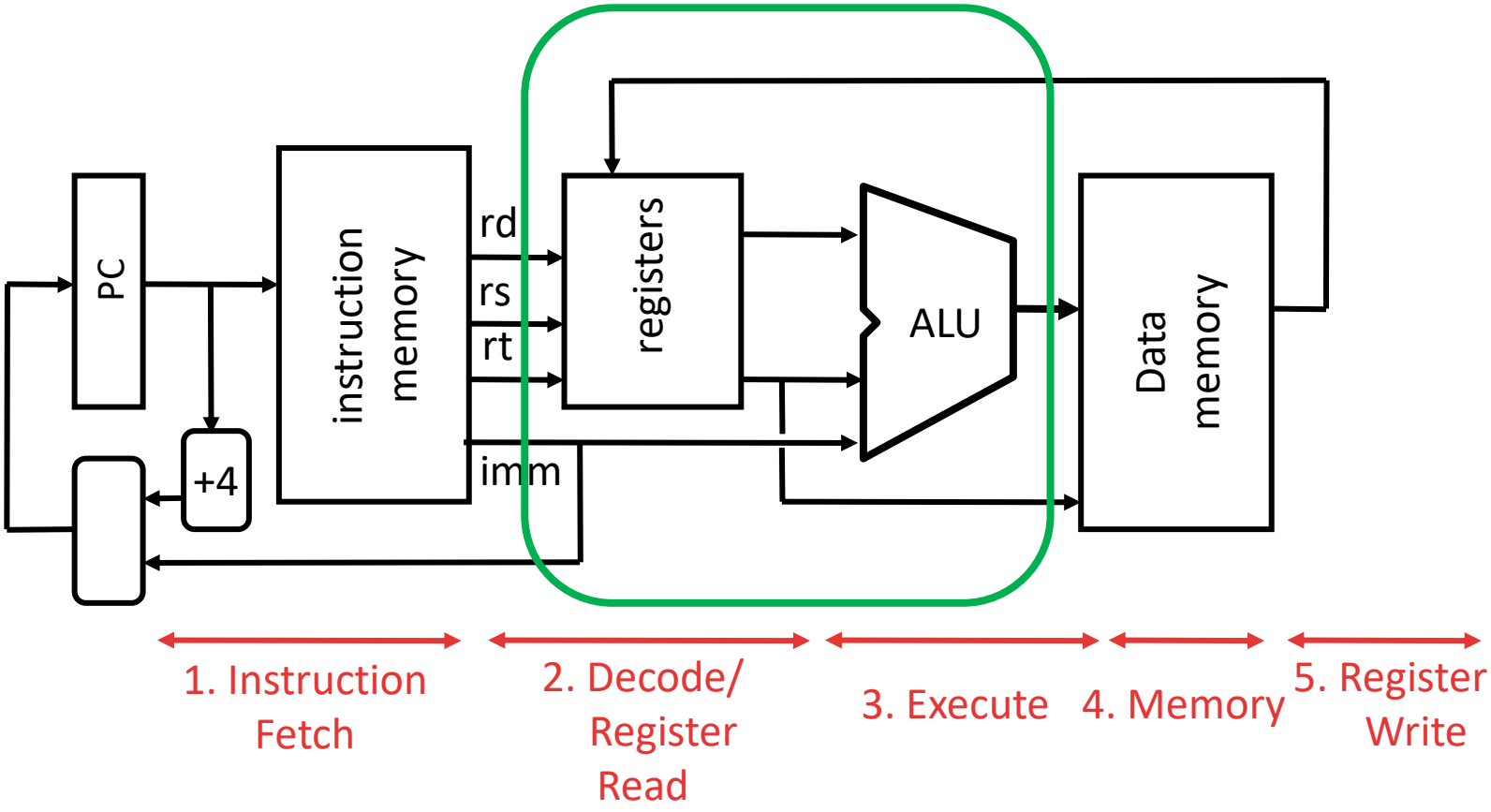
- Labs
  - Lab 10 grace period* ends this week
    - » Demo is **NOT REQUIRED** to receive full credit
- Reading assignments
  - **All** Reading (01-08) and Homework (01-06) assignments **open for submission till 09-MAY, 11:59pm**
    - » Complete **Participation**/**Challenge** Activities in each section to receive grade
    - » IMPORTANT: Make sure to submit score to CatCourses by using the link in the assignment page
    - » You may re-do past Reading/Homework assignments to improve score.

* A 10% penalty will be applied for late *submissions* during the grace period.

Announcements

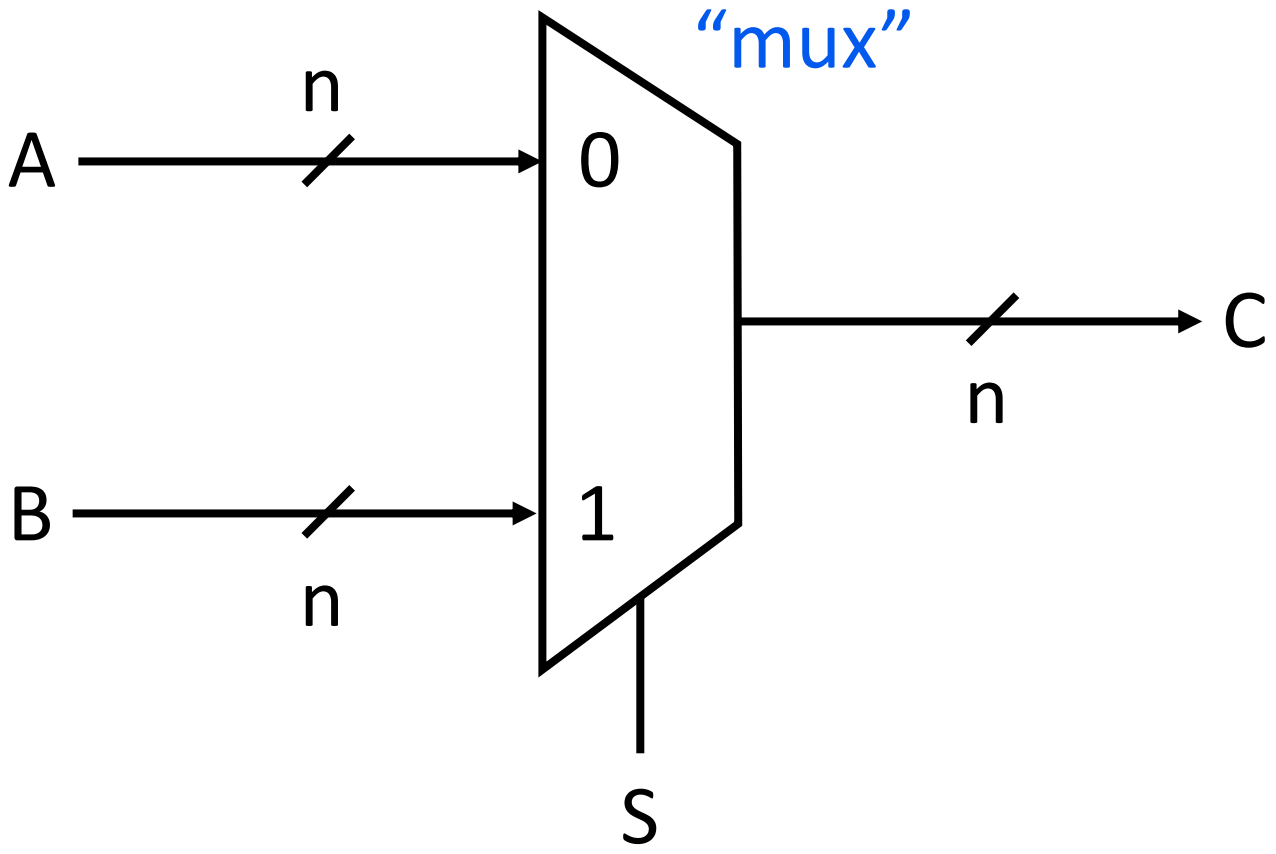
- Project 02
  - Due 05-MAY
  - Can work in teams of 2 students
    - » Each team member must identify teammate in “Comments…” text-box at the submission page
    - » If working in teams, each student must submit code (can be the same as teammate) and demo individually
    - » Grade can vary among teammates depending on demo
  - Demo required for project grade
    - » No partial credit for submission without demo
  - **No grace period**
    - » **Must complete submission and demo by due date.**
- **Extra Credit**
  - Up to 2% towards your overall grades
  - **Due 06-MAY, 01:59am**
  - See assignment page on CatCourses for more details
- Lab with lowest score dropped from final grade evaluation

Generic Steps of Datapath (review)

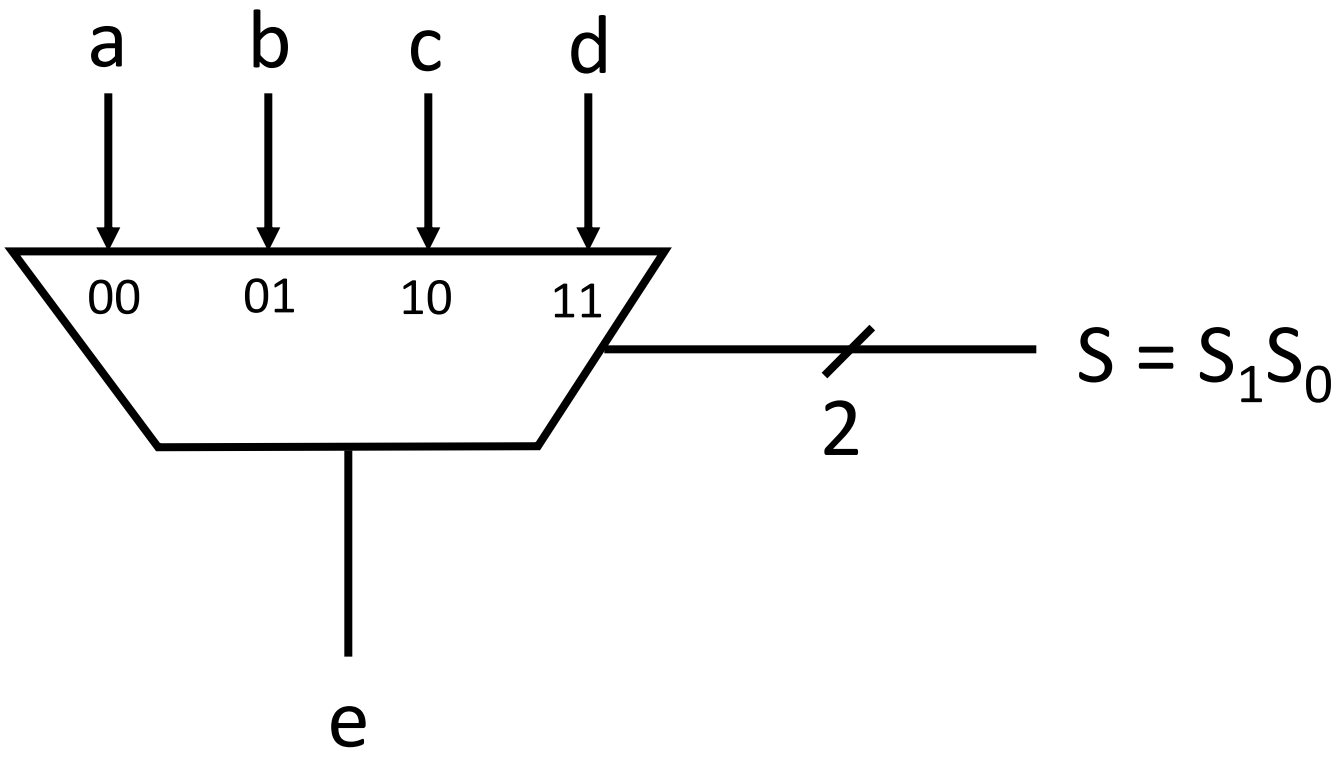


How do we handle the different register usage between r-type and i-type instructions?

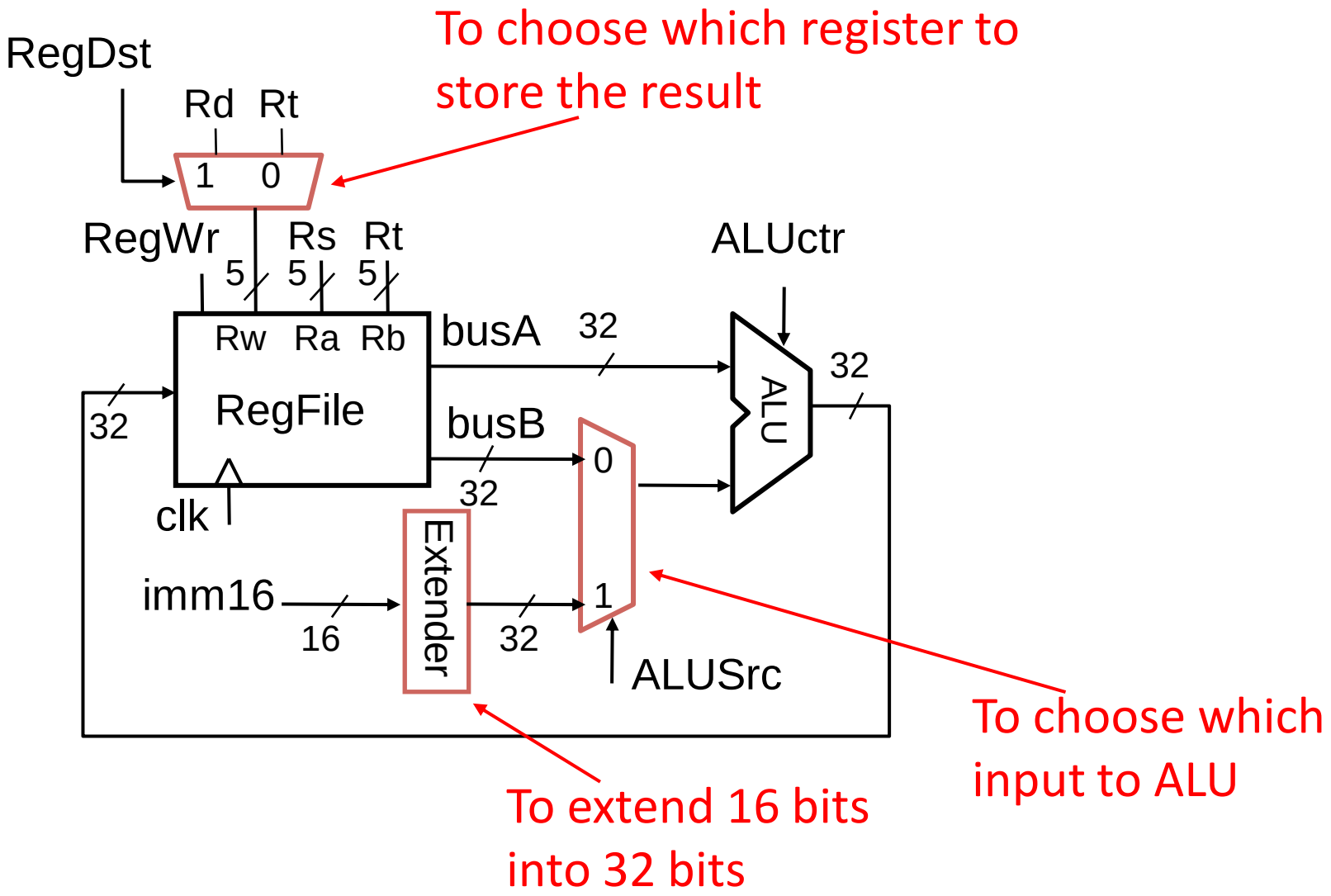
Data Multiplexor or Mux (2-to-1, n-bits)



4-to-1 Multiplexor?

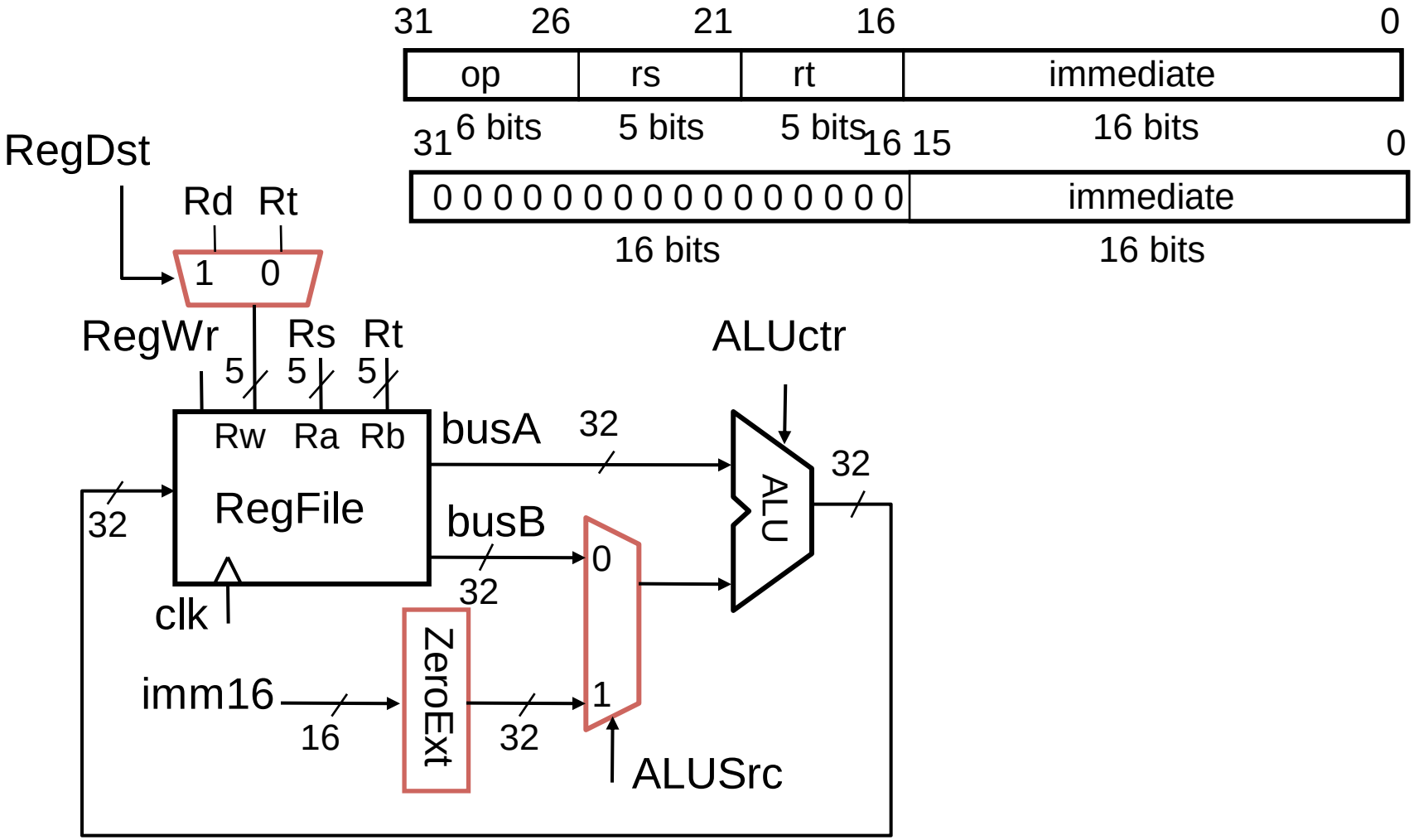


A zoomed in version of RegFile and ALU



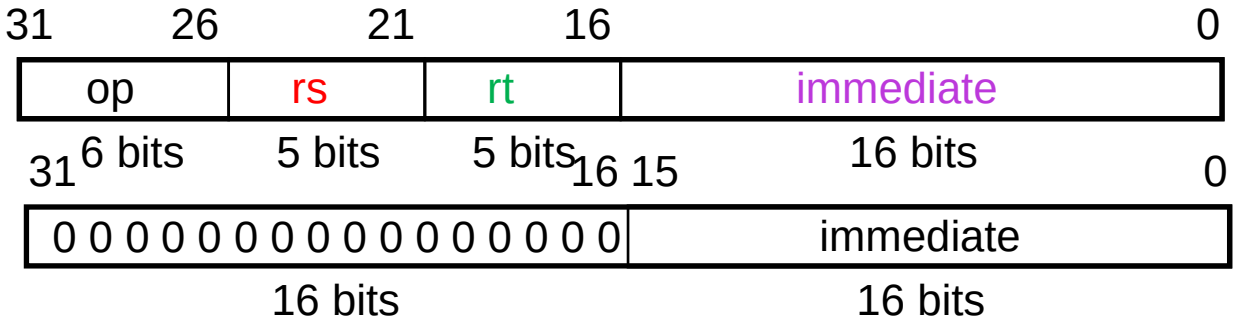
Operations with Immediate

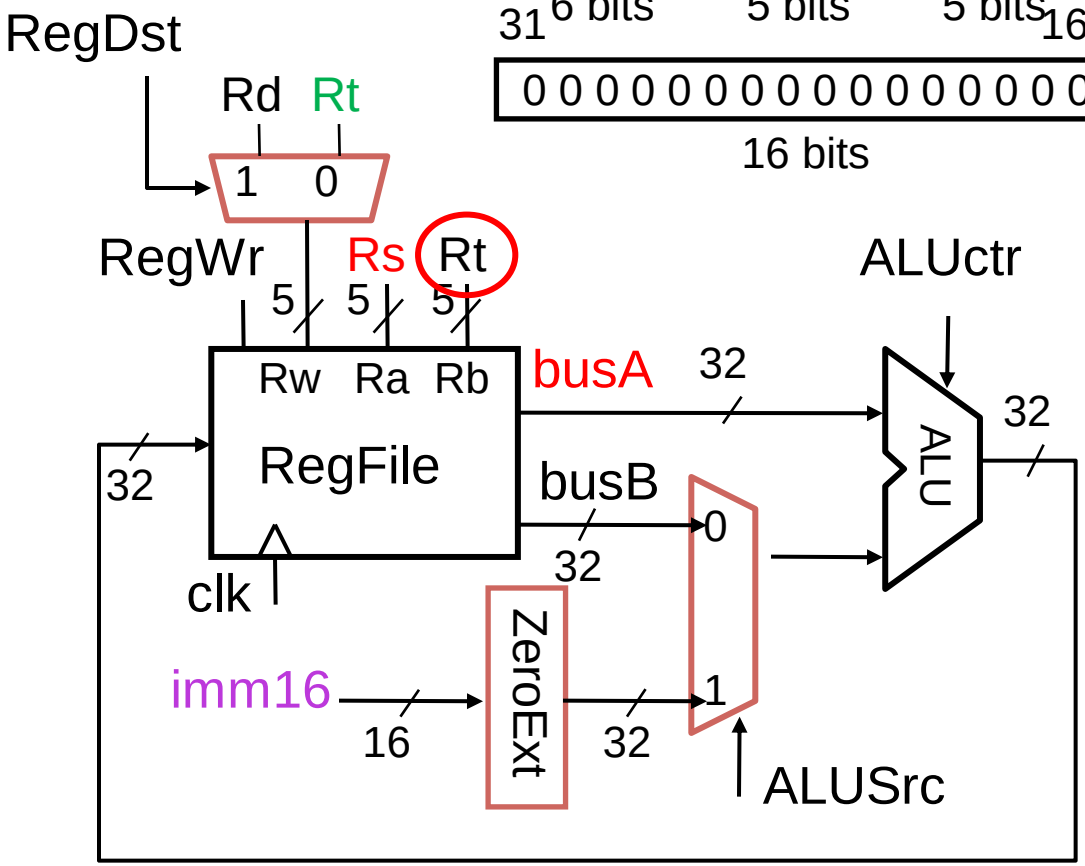
- R[rt] = R[rs] op ZeroExt[imm16]]



Operations with Immediate

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





What about Rt register read (bus B)??

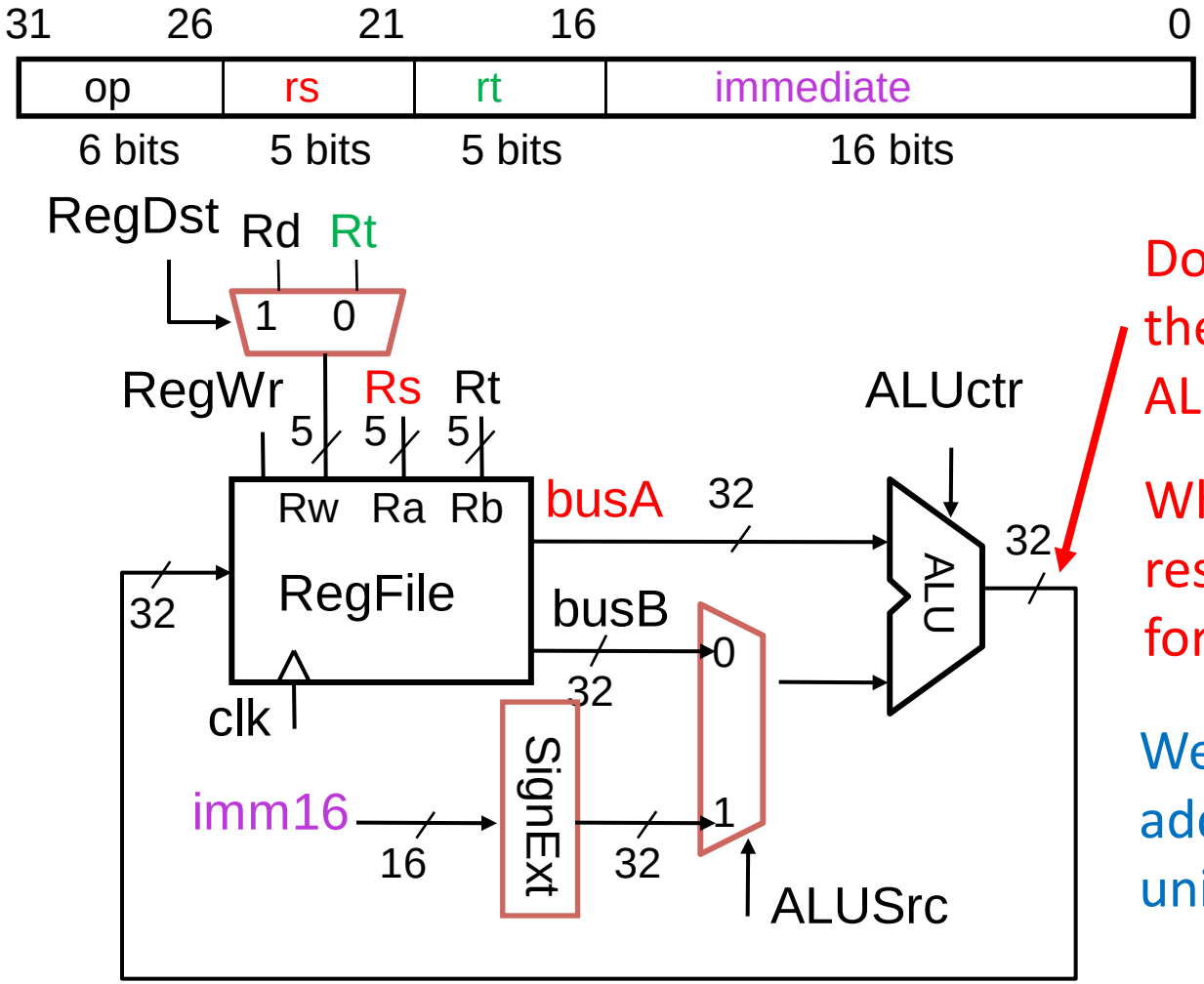
We don't care!

Why?

Bus B is blocked

Load Memory

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



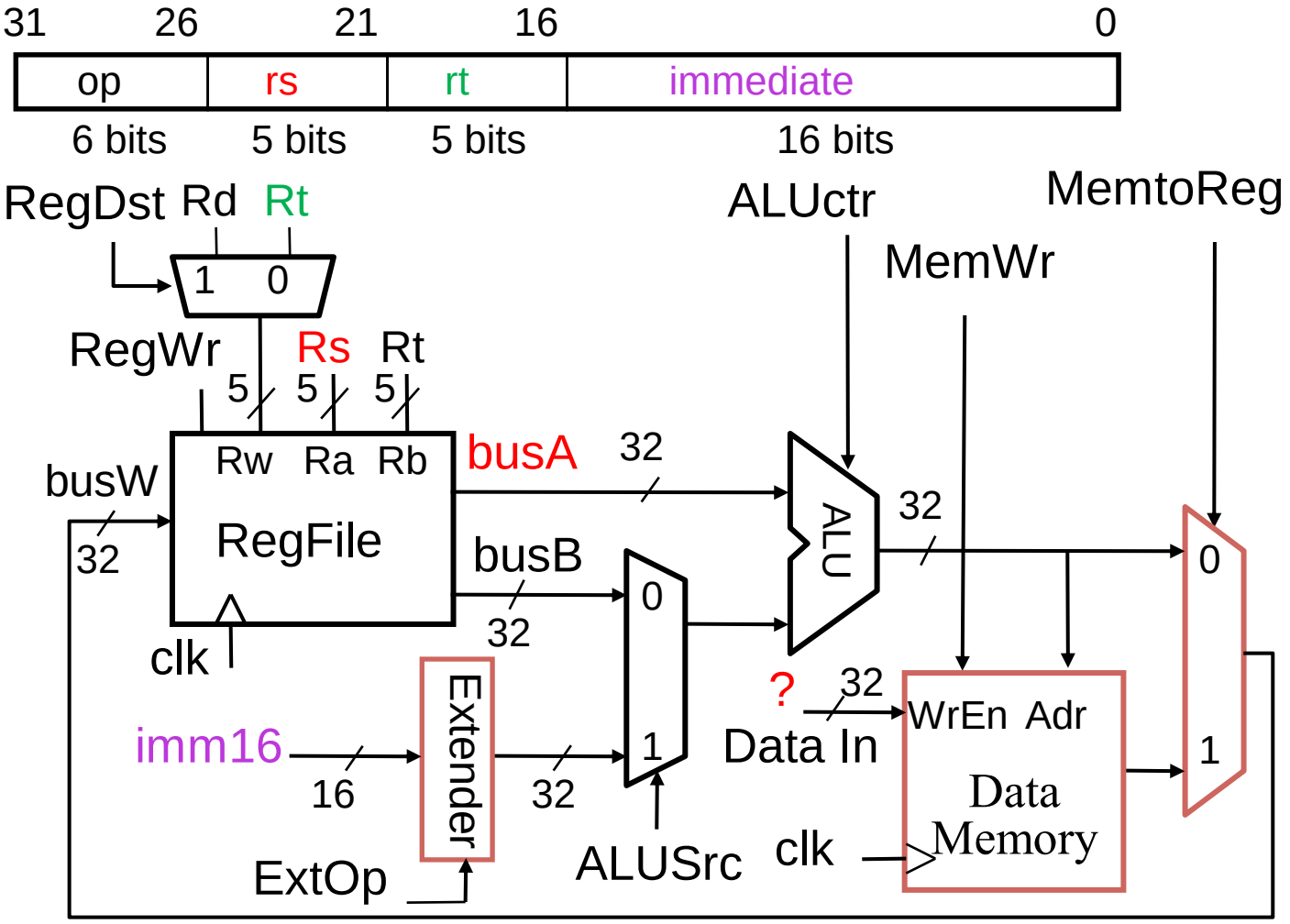
Do we store the result of ALU?

What is the result of ALU for?

We need to add a memory unit!

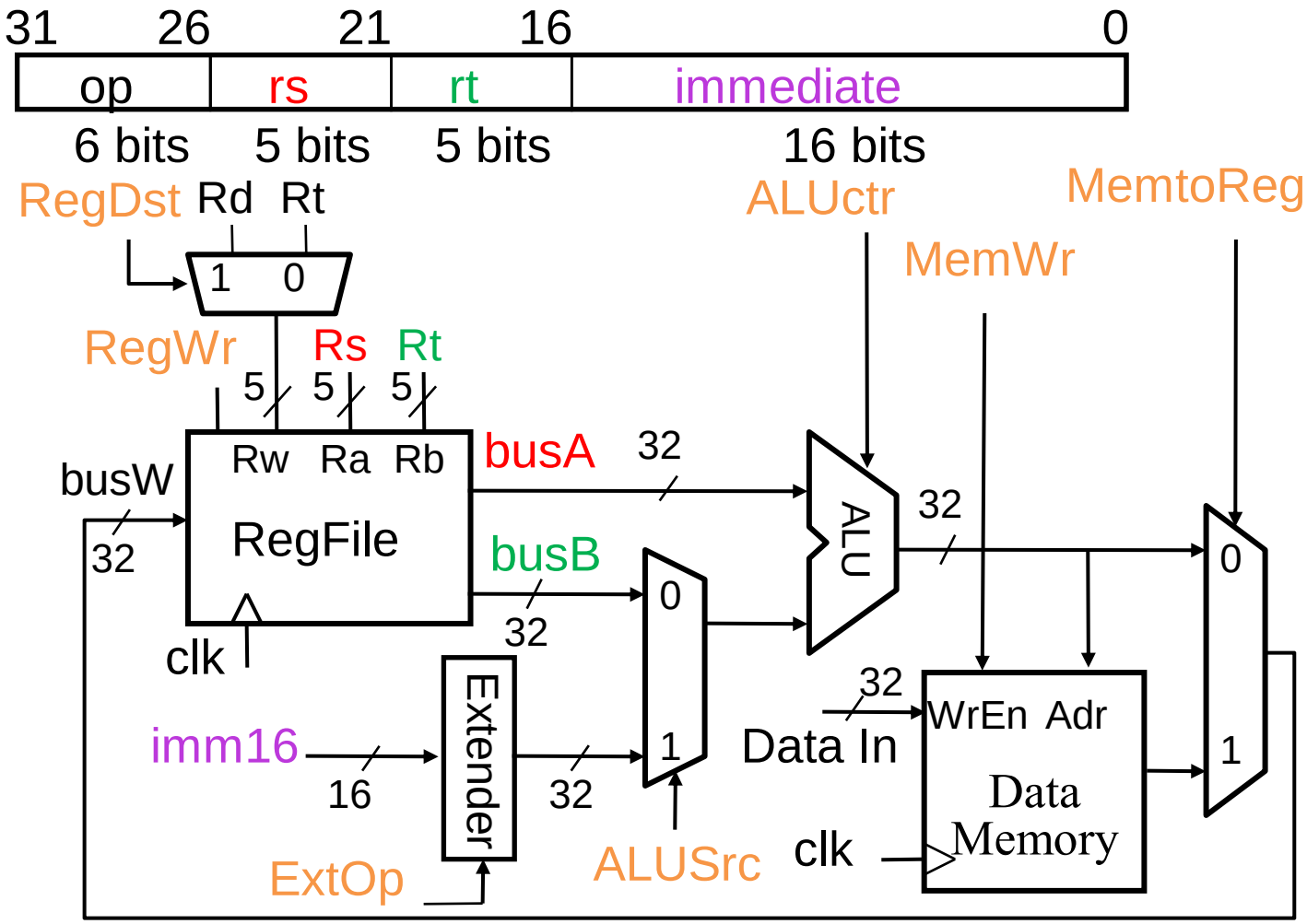
Load Memory

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



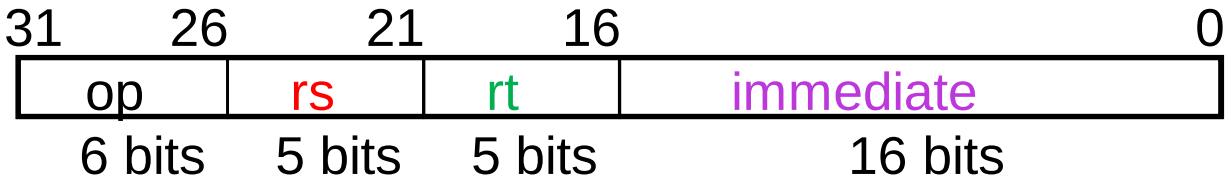
Store Memory

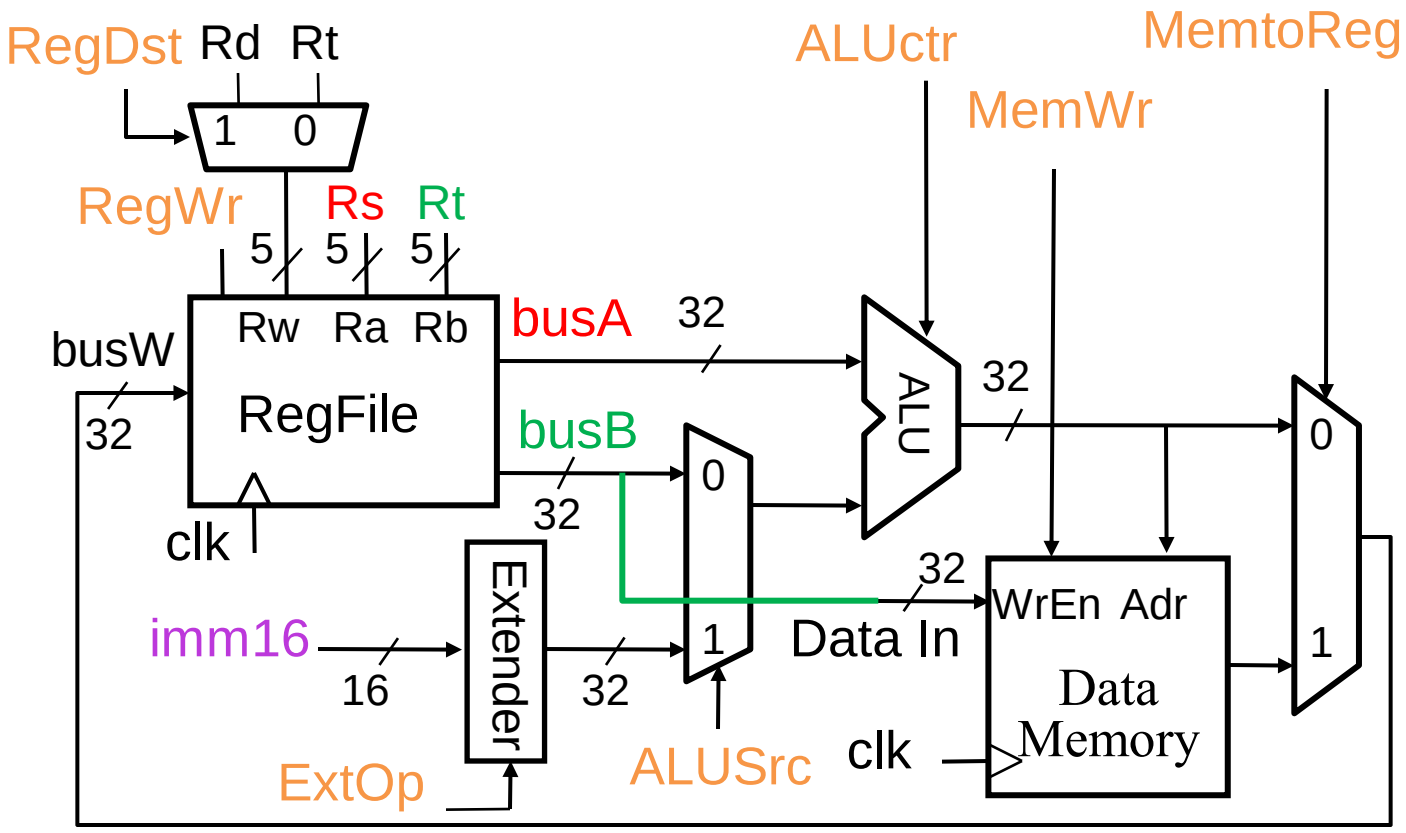
- Mem[R[rs] + SignExt[imm16]] = R[rt]
  Ex.: `sw rt, imm16(rs)`



Store Memory

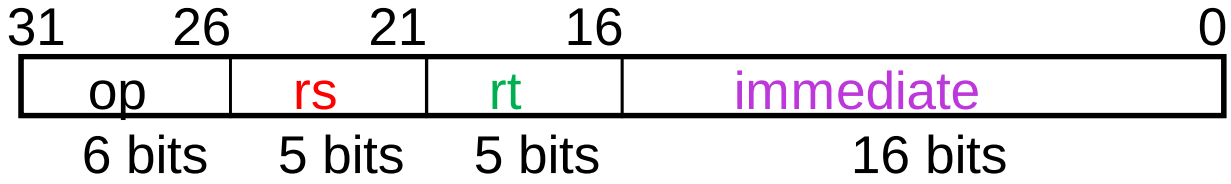
- Mem[R[rs] + SignExt[imm16]] = R[rt]
  Ex.: `sw rt, imm16(rs)`

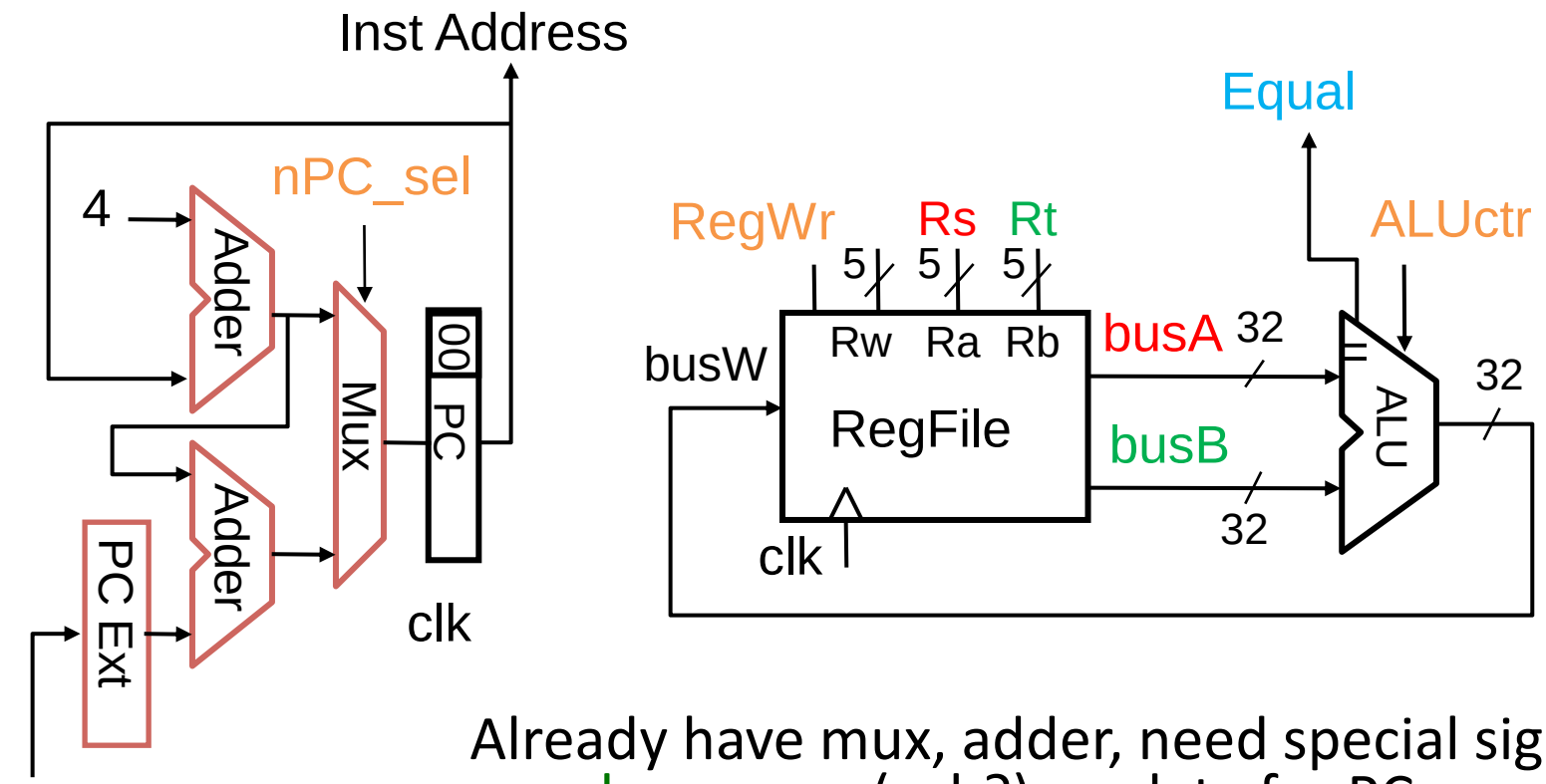




Datapath for Branch Operations

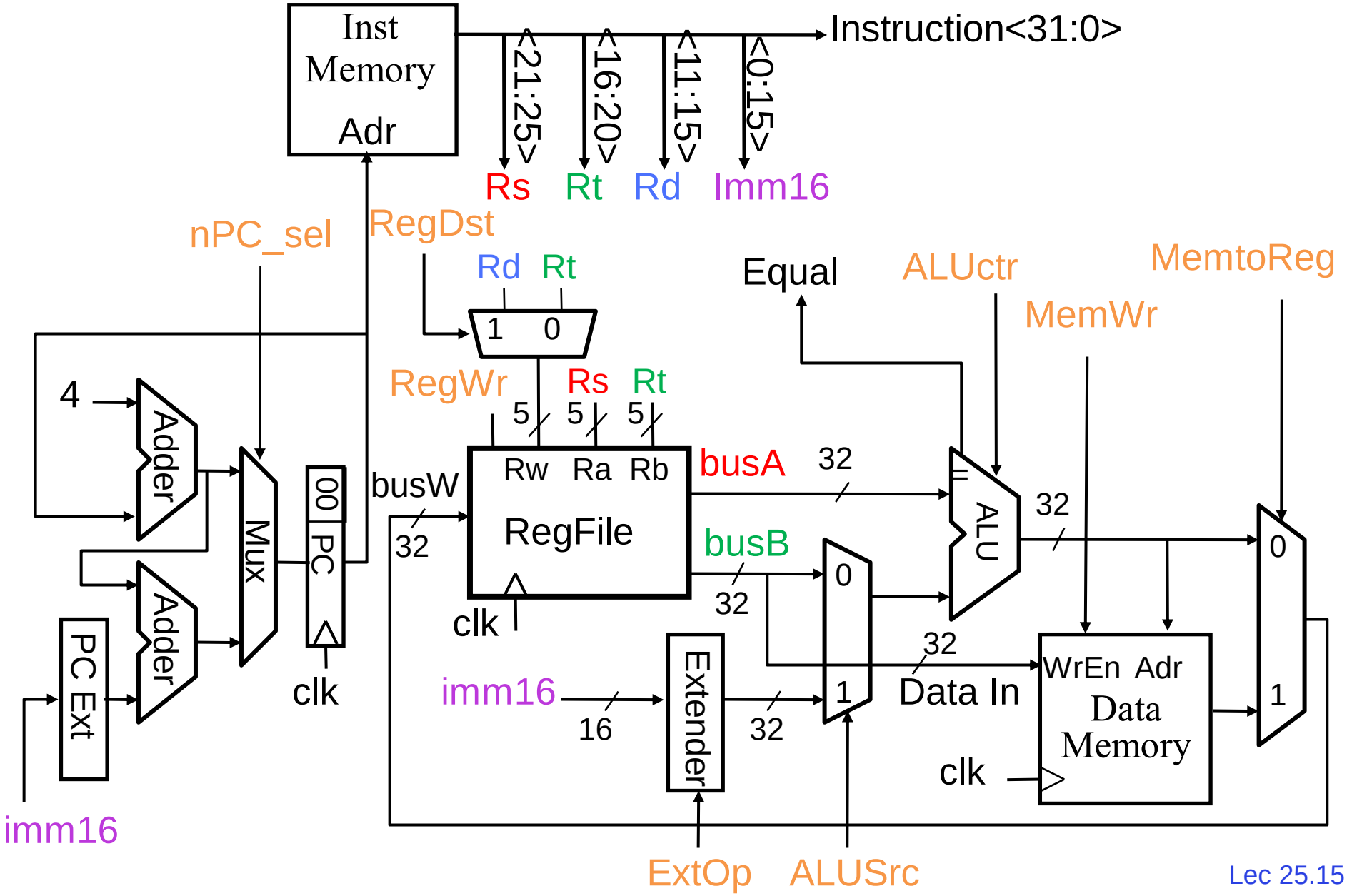
- `beq rs, rt, imm16`
  Datapath generates condition (equal)



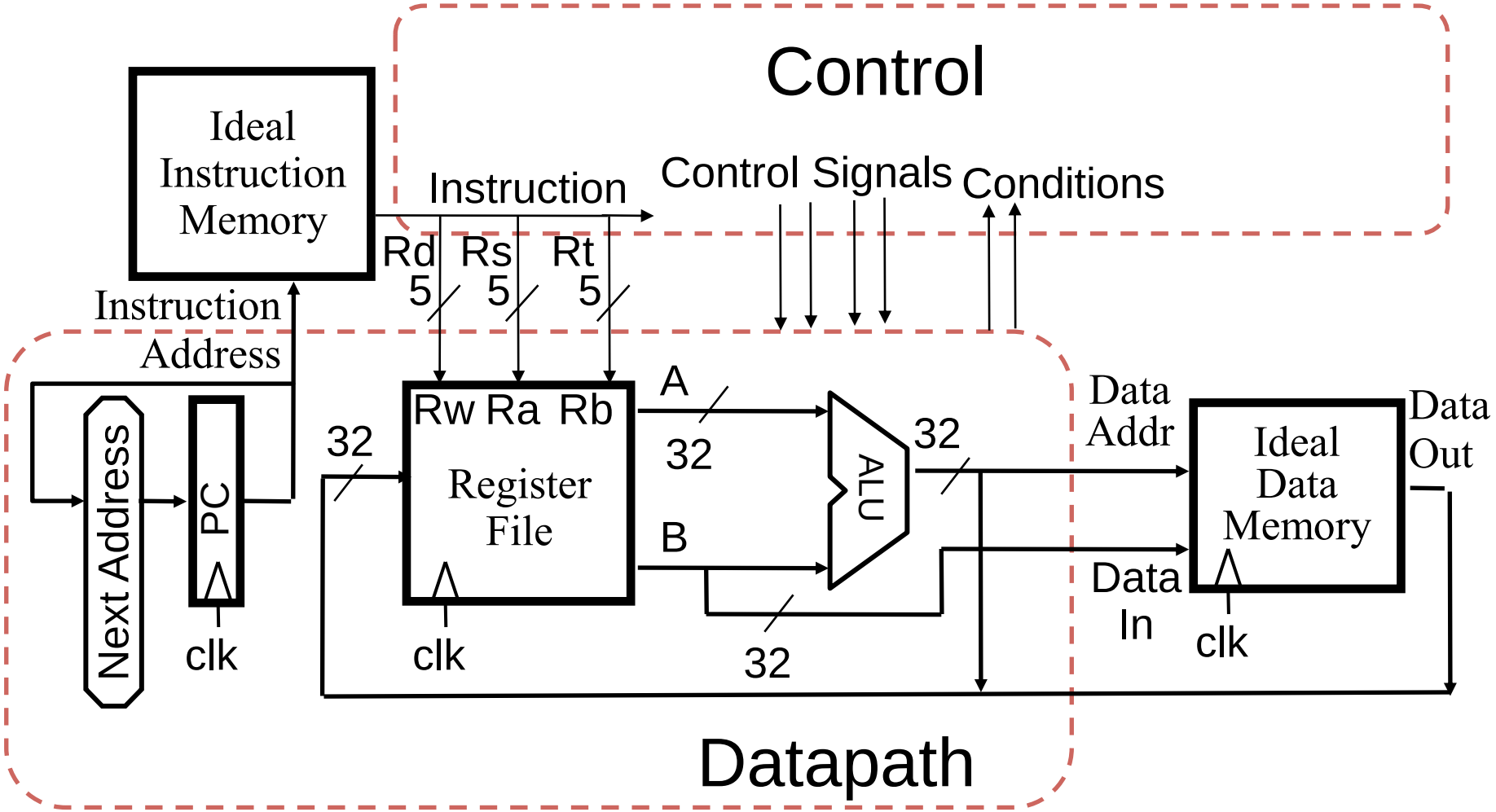


Already have mux, adder, need special sign, need equal compare (sub?), update for PC,

Single Cycle Datapath

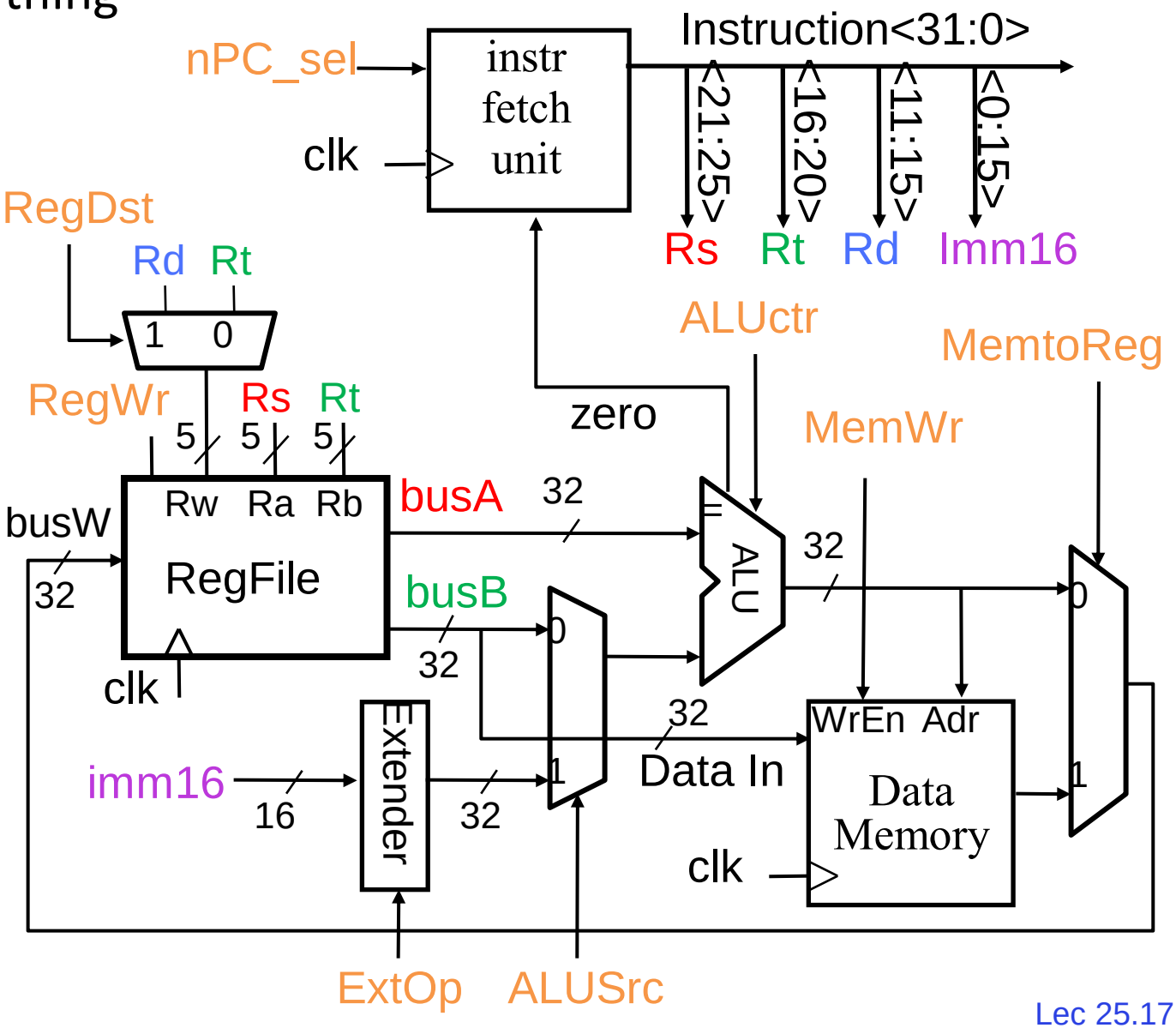


Abstract View of the Implementation



A Single Cycle Datapath

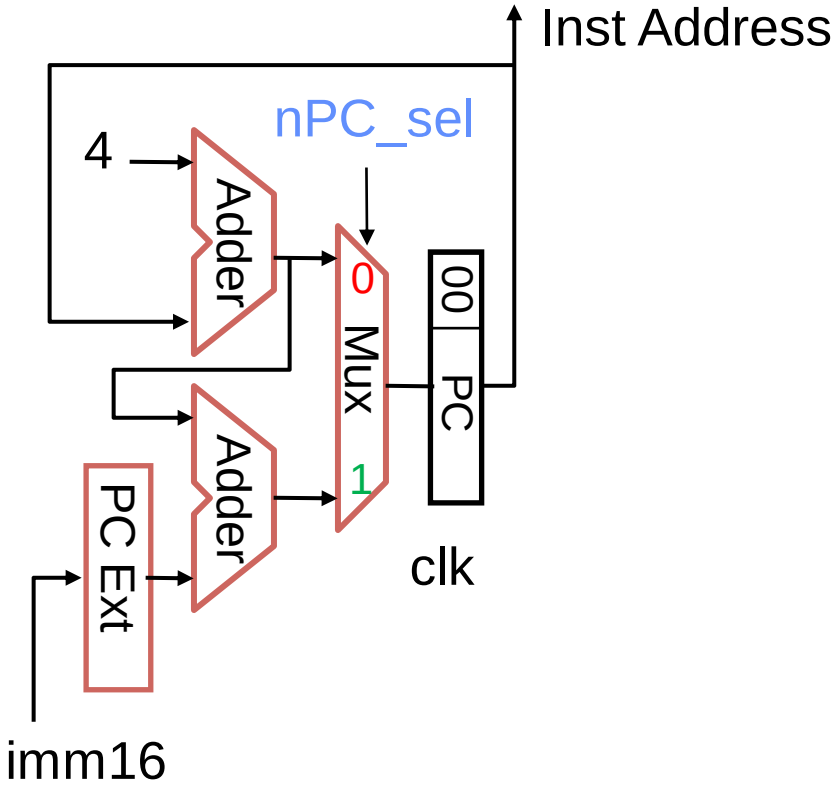
- We have everything except control signals



Meaning of the Control Signals

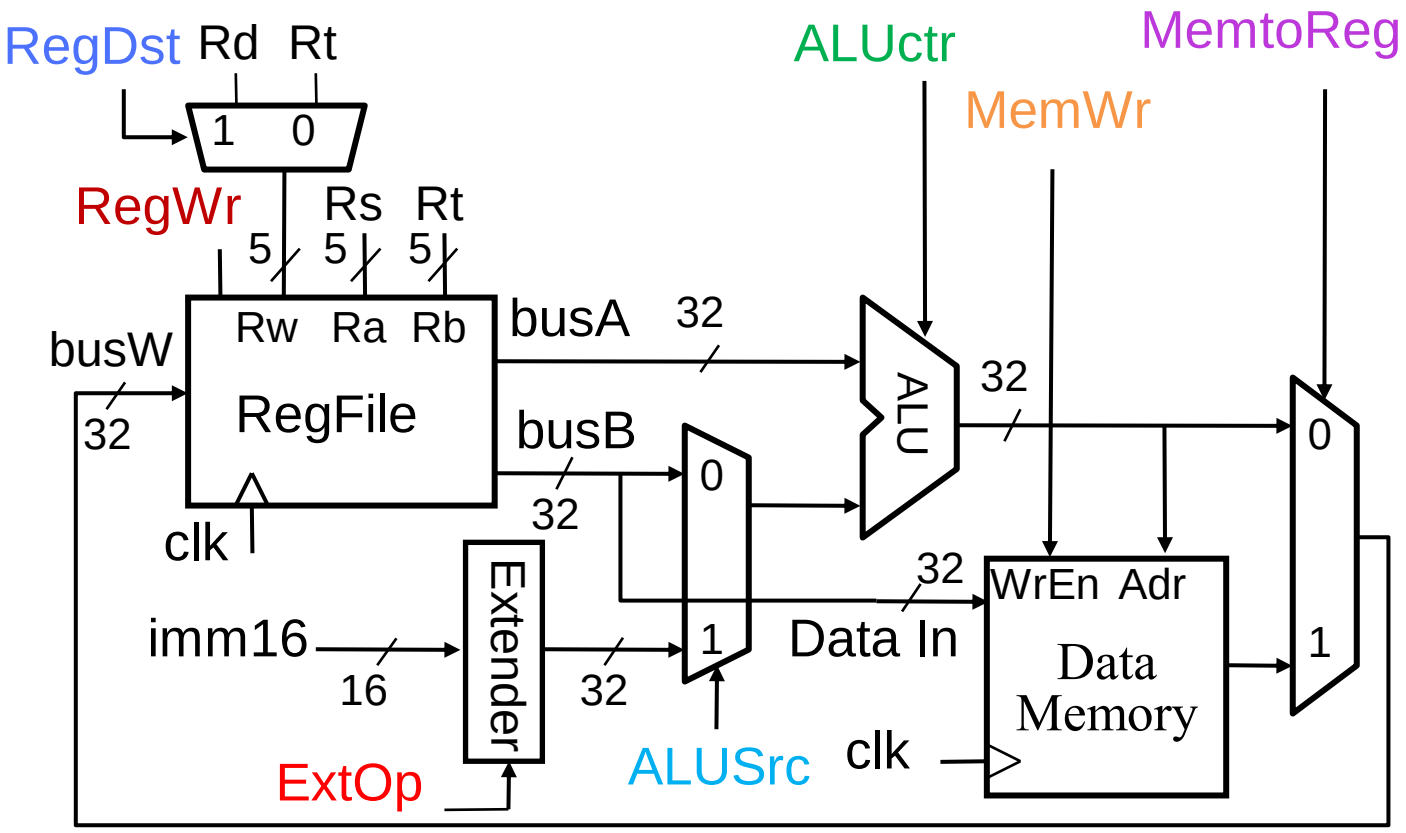
- nPC_sel: "+4": 0 ⇒ PC <– PC + 4
  "br": 1 ⇒ PC <– PC + 4 + {SignExt(Im16) , 00 }
  "n"=next

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

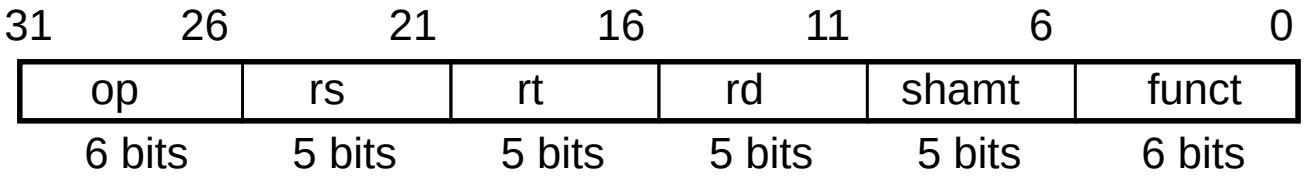


Meaning of the Control Signals

- ExtOp: 0 ⇒ zero extend
  1 ⇒ sign extend
- 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



`add rd, rs, rt`

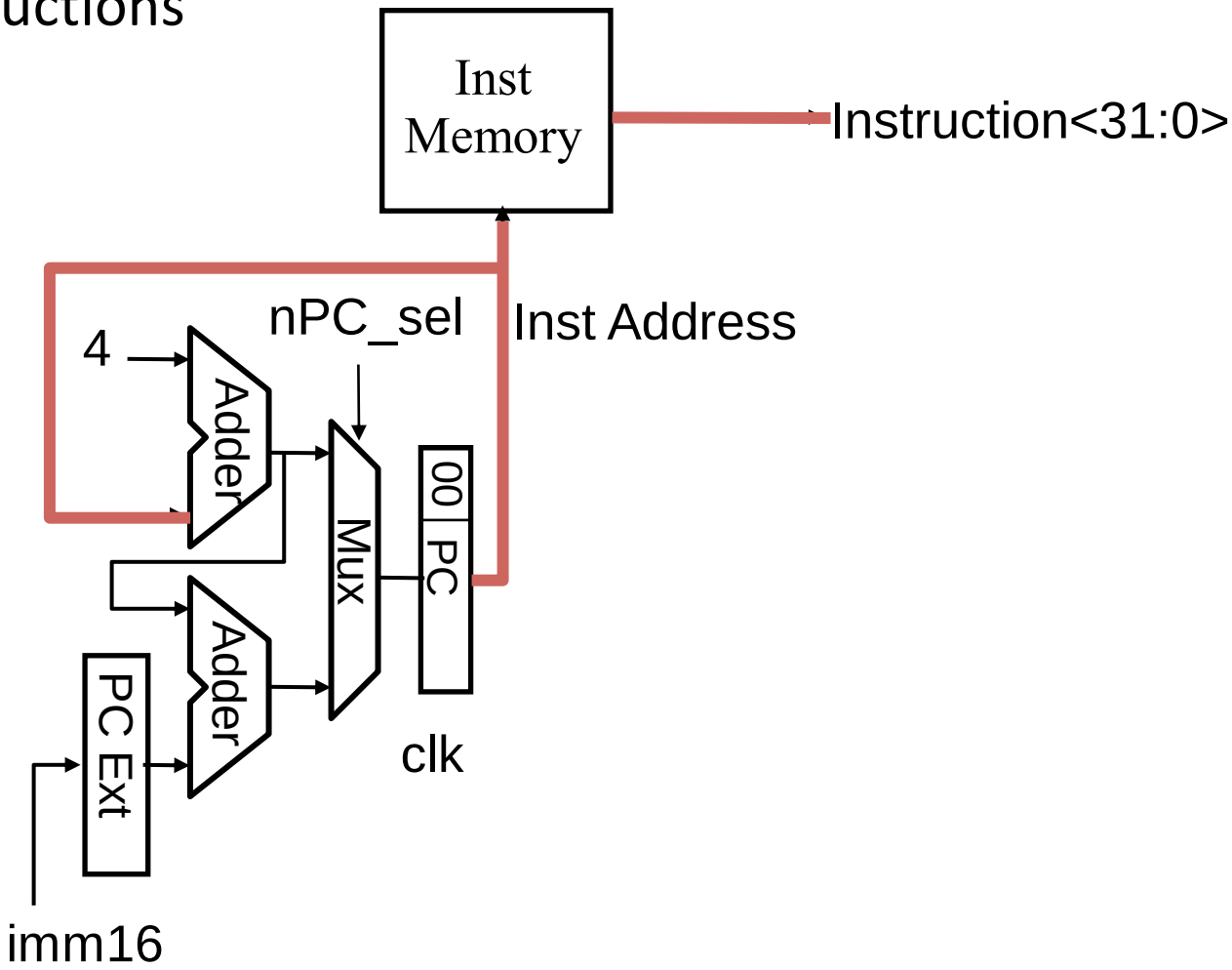
- MEM[PC] — Fetch the instruction from memory
- R[rd] = R[rs] + R[rt] — The actual operation
- PC = PC + 4 — Calculate the next instruction's address

Instruction Fetch Unit start of Add

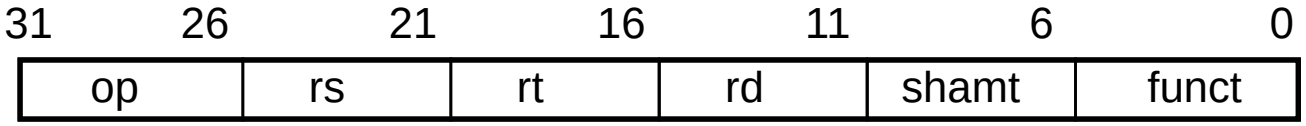
- Fetch the instruction from Instruction memory:

  Instruction = MEM[PC]

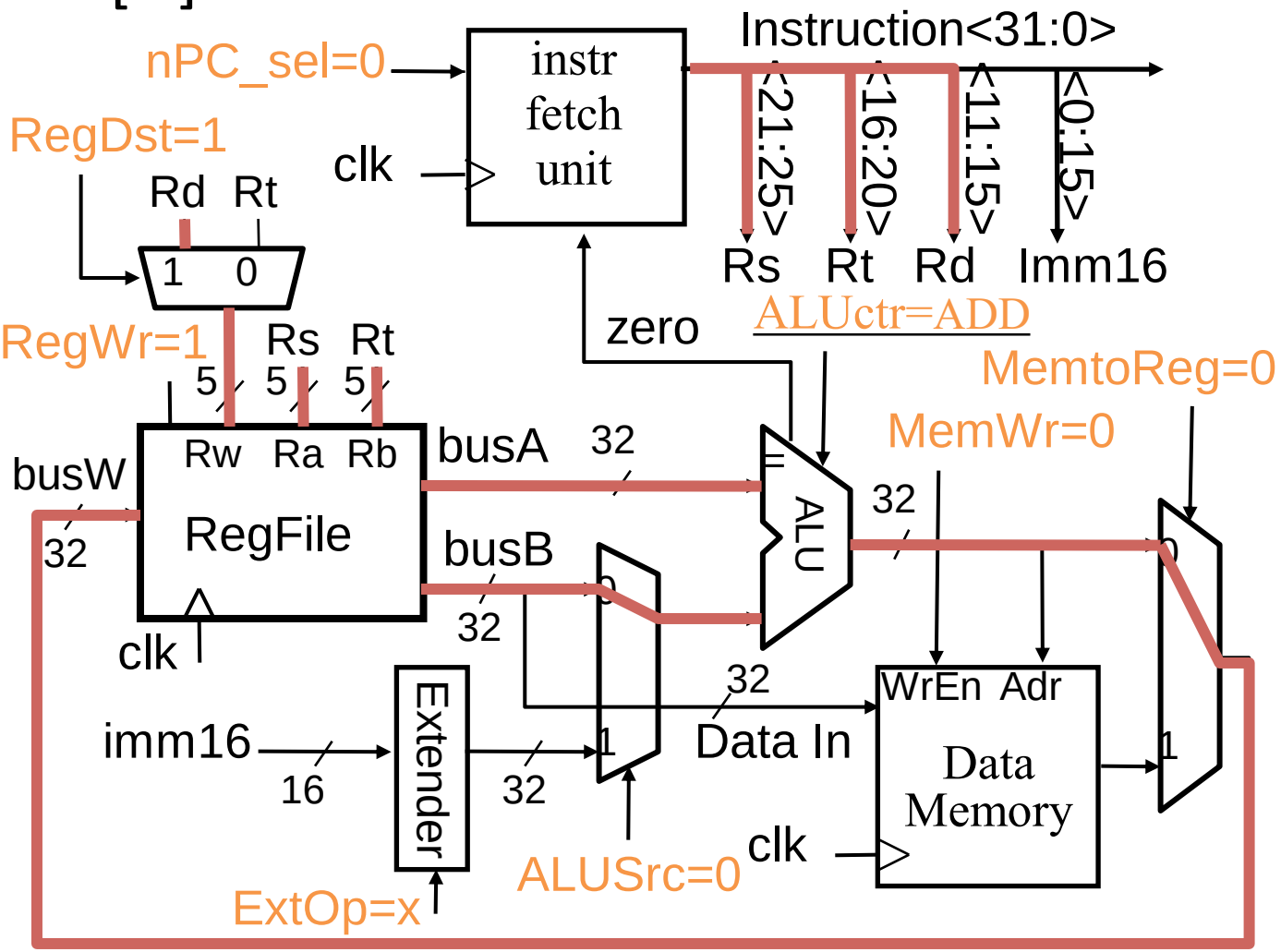
  – same for all instructions



The Single Cycle Datapath during Add

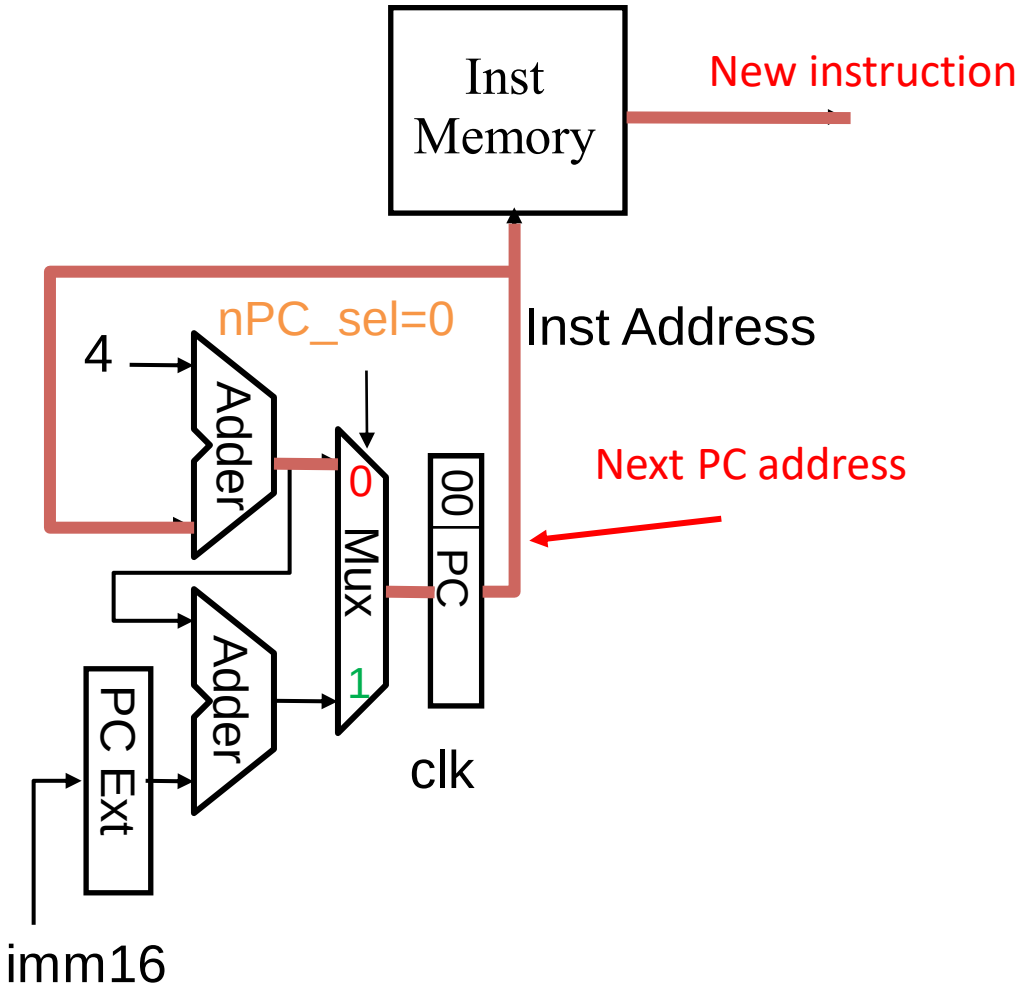


R[rd] = R[rs] + R[rt]

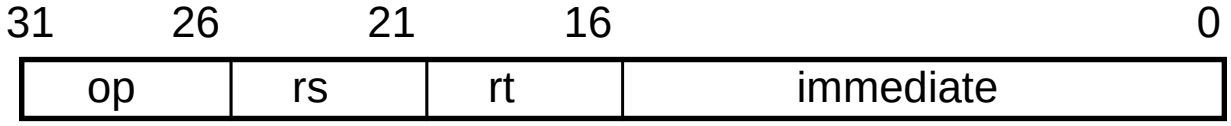


Instruction Fetch Unit end of Add

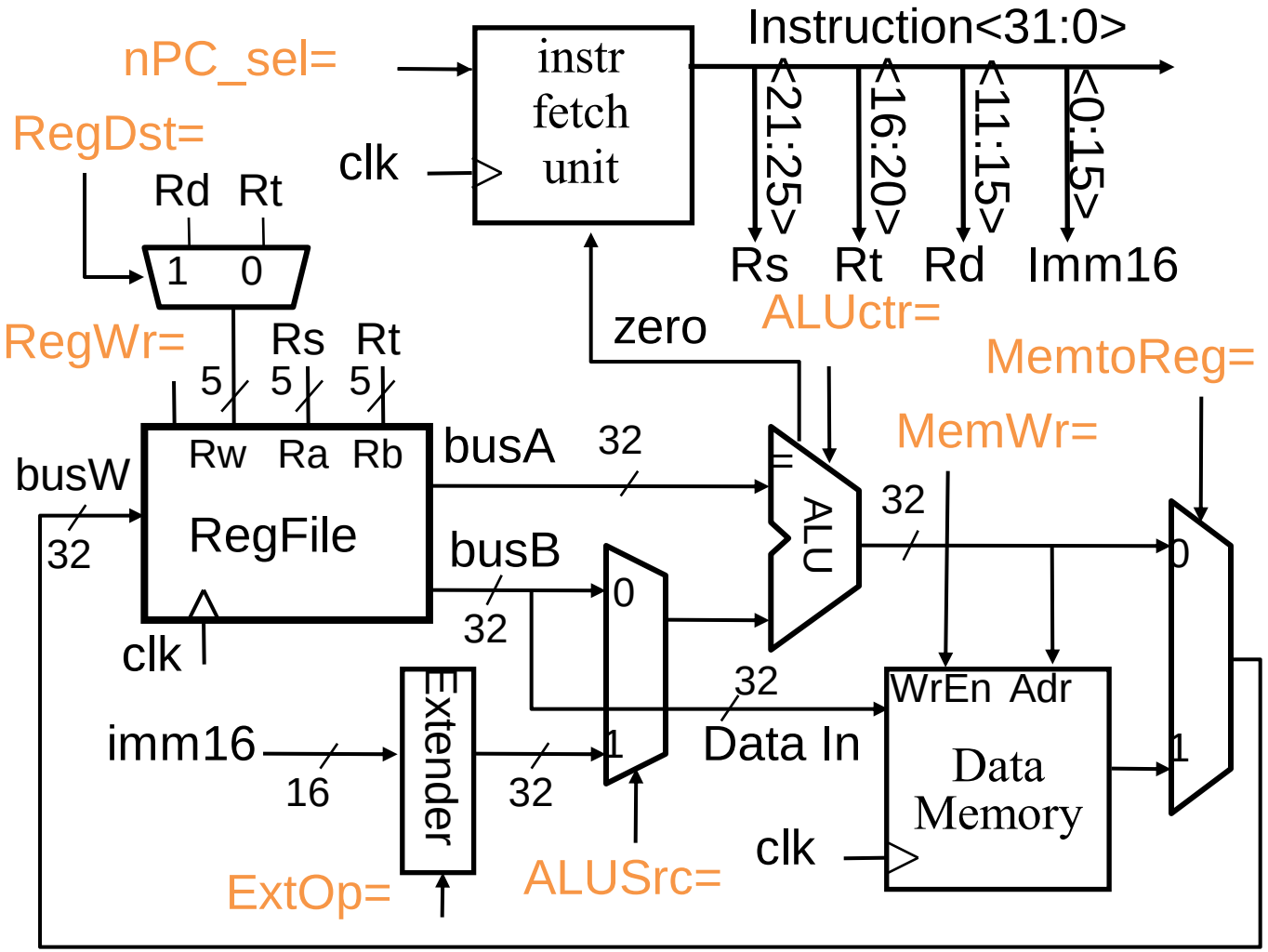
- PC = PC + 4
  - This is the same for all instructions except: Branch and Jump



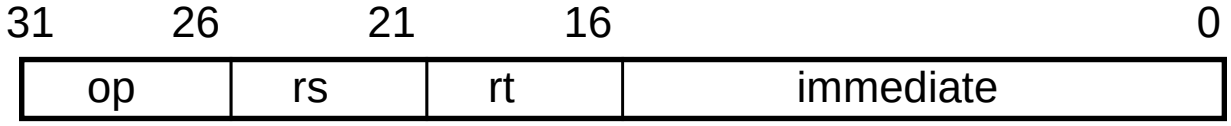
Single Cycle Datapath for Ori



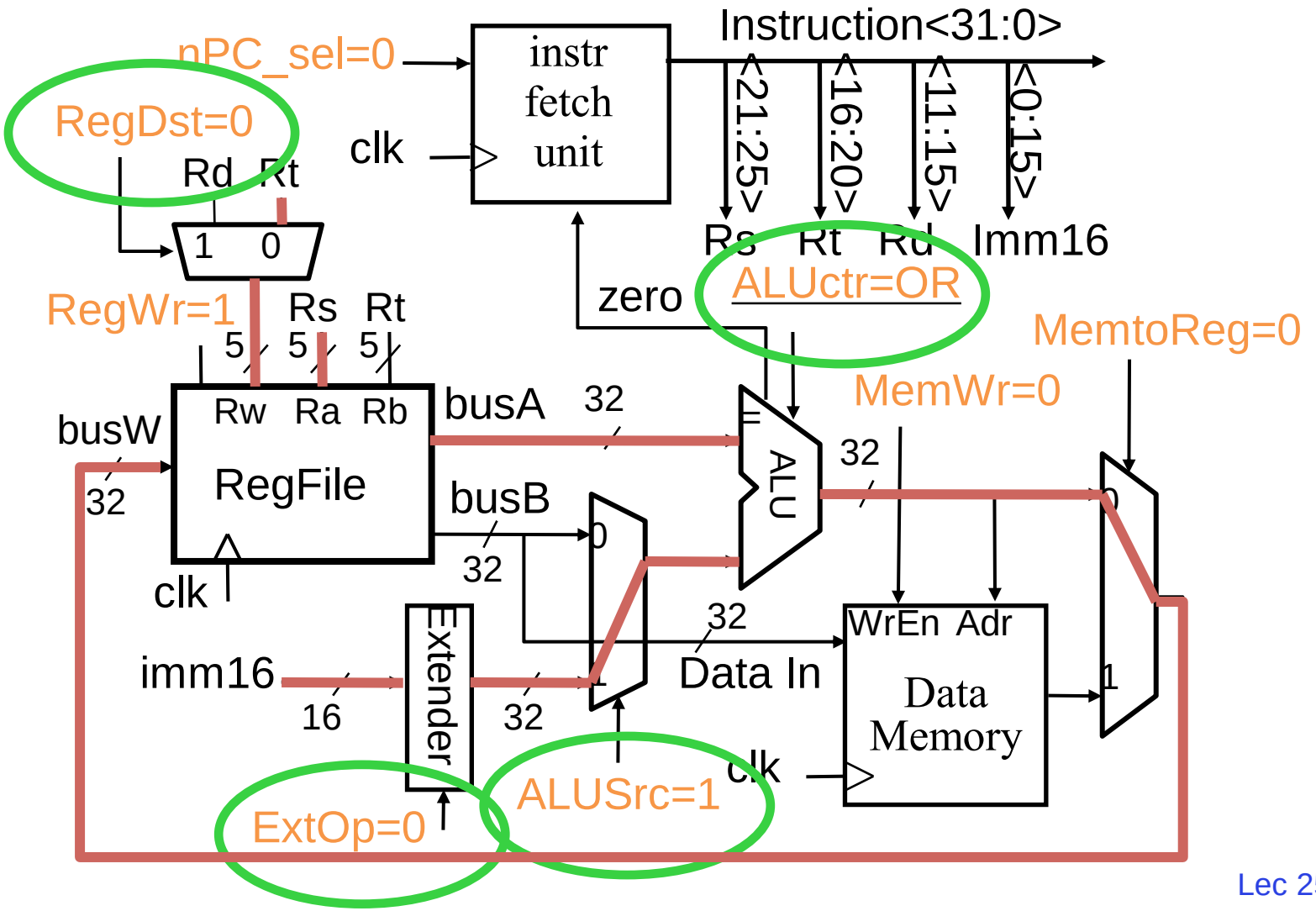
- R[rt] = R[rs] OR ZeroExt[Imm16]



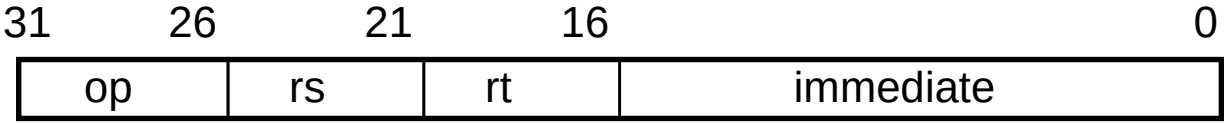
Single Cycle Datapath for Ori



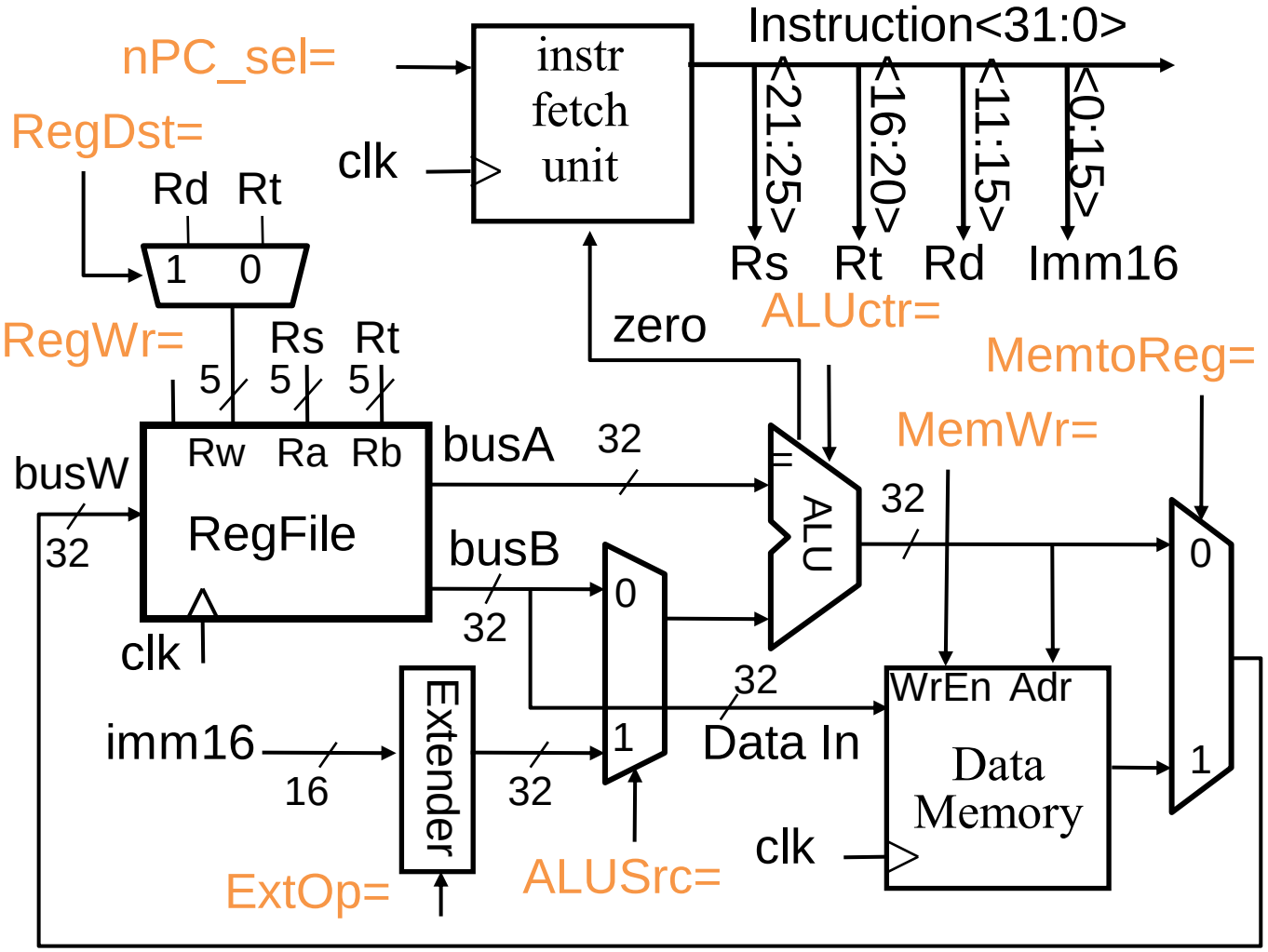
- R[rt] = R[rs] OR ZeroExt[Imm16]



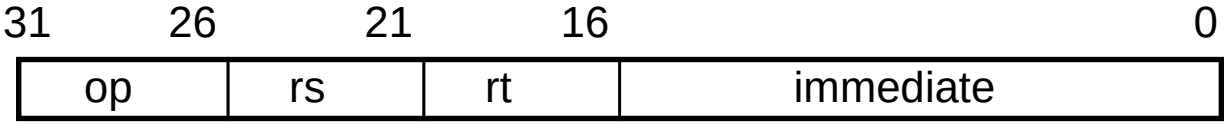
Single Cycle Datapath for LW



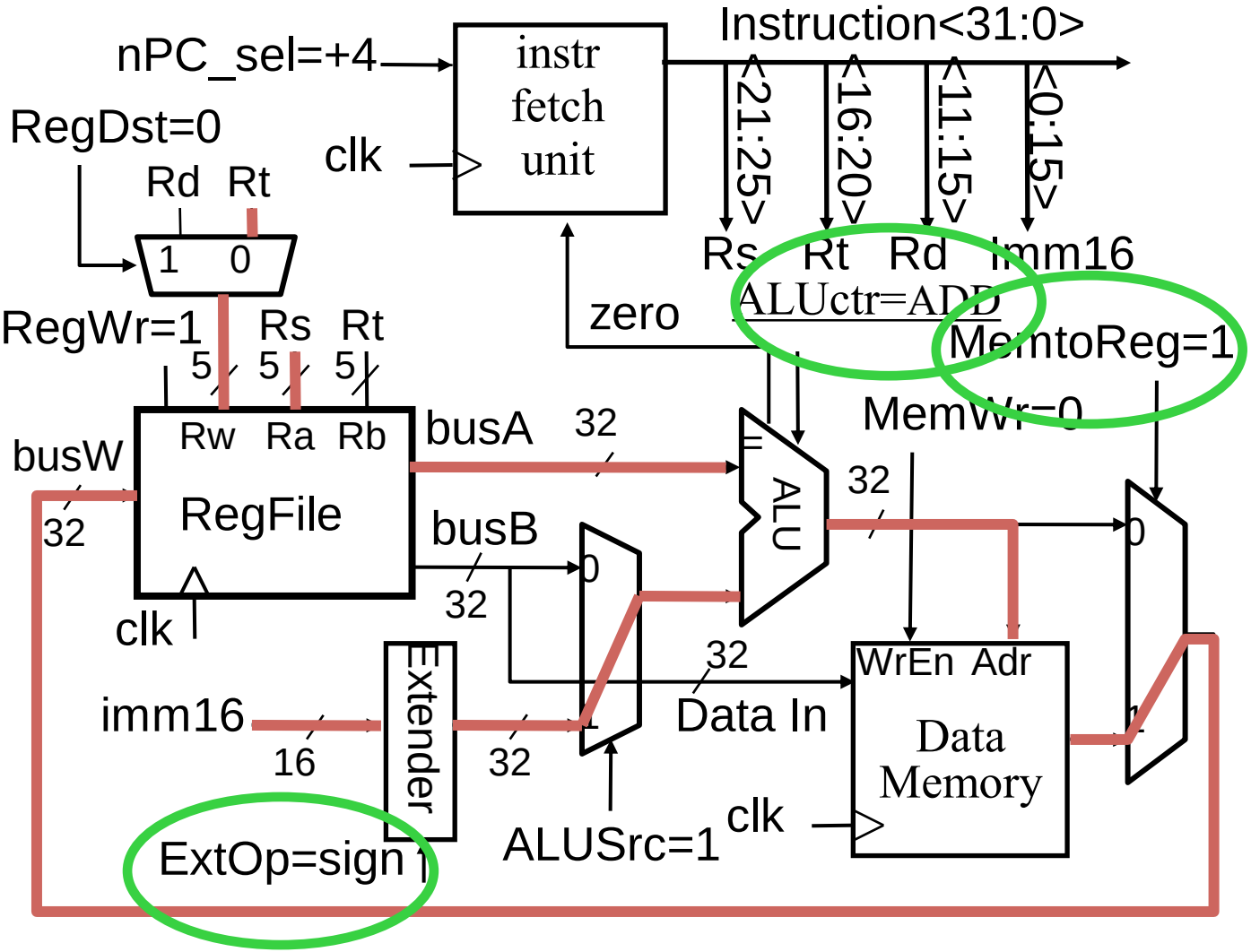
- R[rt] = Data Memory {R[rs] + SignExt[imm16]}



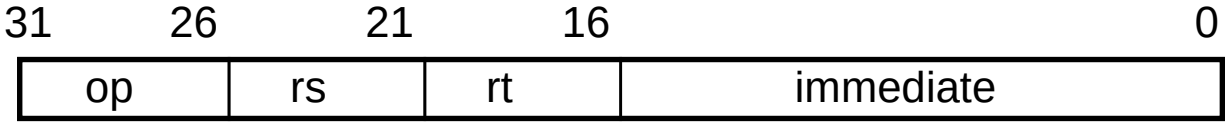
Single Cycle Datapath for LW



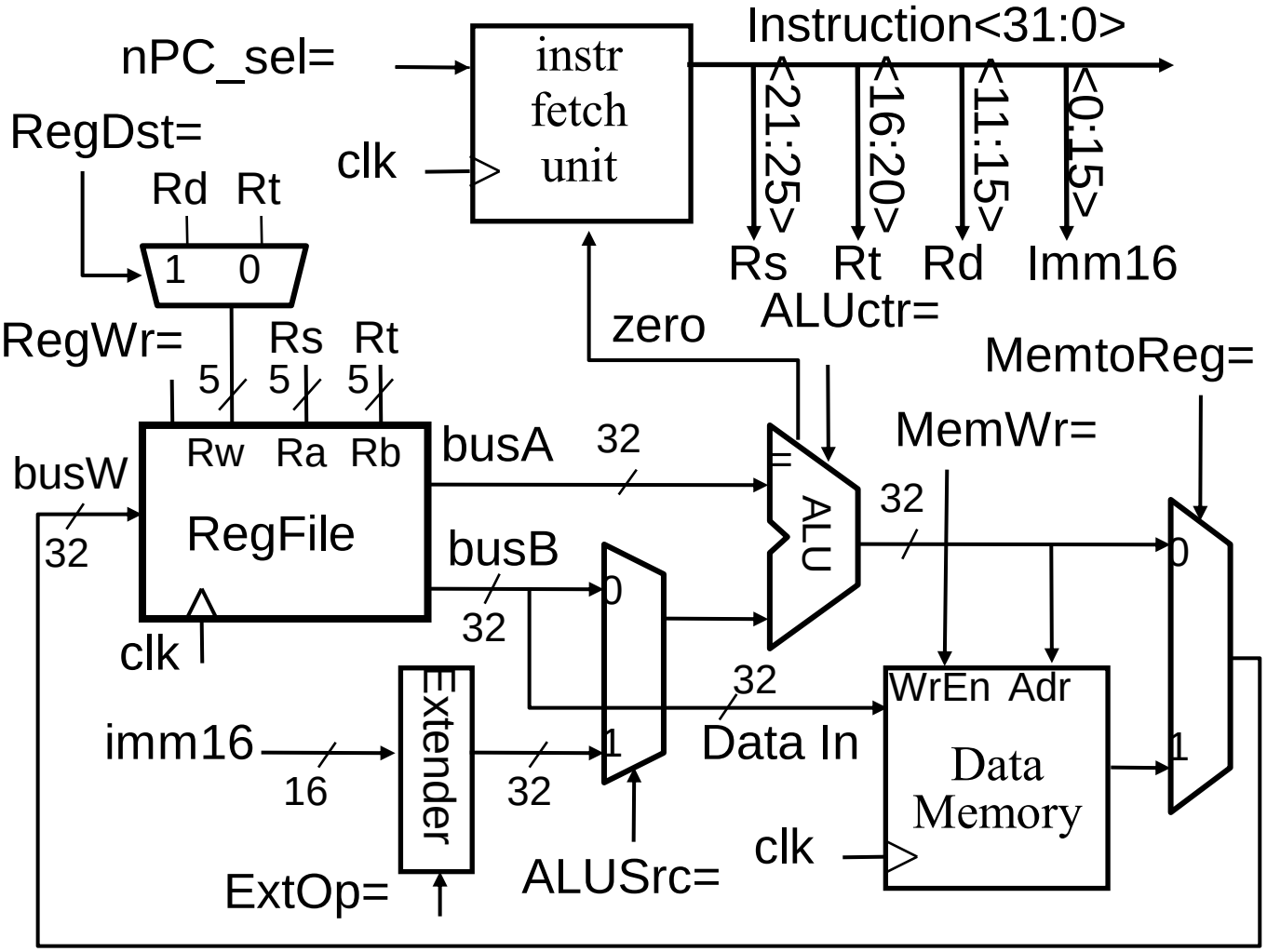
- R[rt] = Data Memory {R[rs] + SignExt[imm16]}



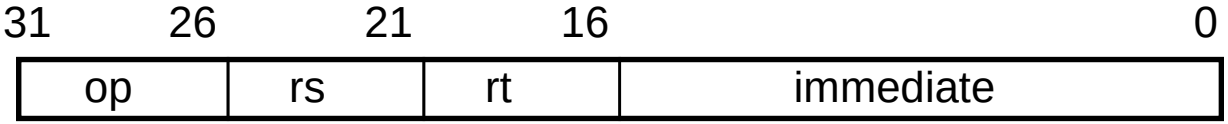
Single Cycle Datapath for SW



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



Single Cycle Datapath for SW



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