EGC442
Class Notes
3/10/2023

Baback Izadi
Division of Engineering Programs
bai@engr.newpaltz.edu

Test 1

Average	82.5
Median	85.5
MAX	92.0
Minimum	67.0

Consider a rising clock edge that causes 3000 to be written into the PC.

1) The 3000 waits at the instruction memory input for the next rising clock edge, at which time the instruction at address 3000 is read out.
   - ○ True
   - ◉ False

Correct

Because the instruction memory only reads, the instruction memory is like combinational logic. So the read begins as soon as the new address arrives, without waiting for a rising clock edge.

2) After the address 3000 is read into the PC, the 3000 only propagates to the adder.
   - ○ True
   - ◉ False

Correct

When the 3000 is written into the PC, the 3000 propagates simultaneously to both the instruction memory and the adder.

3) The 3000 waits at the adder input for the next rising clock edge.
   - ○ True
   - ◉ False

Correct

The adder is combinational logic, so the 3000 enters the adder logic without waiting for a rising clock edge.

4) 3001 will be waiting at the PC's input to be written on the next instruction fetch cycle.
   - ○ True
   - ◉ False

Correct

The adder adds 4, not 1, because each MIPS word is 4 bytes.

1) The register file always outputs the two registers' values for the two input read addresses.

- (●) True
- ( ) False

Correct

The register file does not wait for a rising clock edge, nor any special control signals, to output those two registers' values.

2) The register file writes to one register on every rising clock edge.

- ( ) True
- (●) False

Correct

The write only occurs if the RegWrite input is 1.

3) The design can read from two registers and write to one register during the same clock cycle.

- (●) True
- ( ) False

Correct

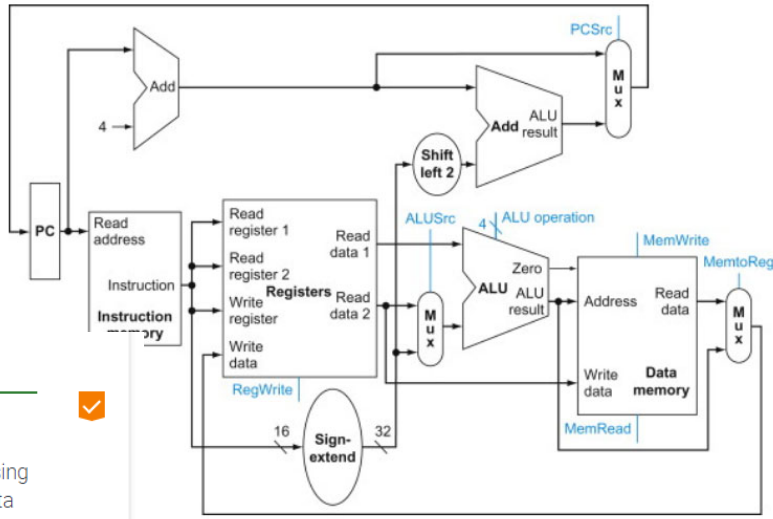
Because the design is edge triggered, the read values will be waiting at register inputs to be written on the next rising clock edge.

4) The programmer must take care not to create a program that writes to a register during the same cycle that the same register is read.

- ( ) True
- (●) False

Correct

Instructions like add $s1, $s1, $s0 (read $s1 and $s0, add values, and write the result to $s1) are common. Because the design is edge triggered, the reads will occur, then the add, and the result will be waiting and ready to be written on the next rising clock edge.



Consider the MIPS datapath above. Find the error in each of the following statements.

1) An R-type instruction like add uses three datapath units: the register file, the ALU, and the data memory.

Correct

Such an instruction reads register values, operates using the ALU, and writes a result back to a register. The data memory is not involved (no load or store is occuring).

2) In addition to the register file and ALU, a load or store instruction also involves the sign extension, data memory, and shift left units.

Correct

The shifter is used by branch instructions, but is not needed by load or store instructions.

3) The branch datapath uses the sign extension, shift by 2, and data memory units.

Correct

The data memory is not involved in a jump instruction.

4) During a load instruction, the mux to the data memory's right would pass the top / bottom input.

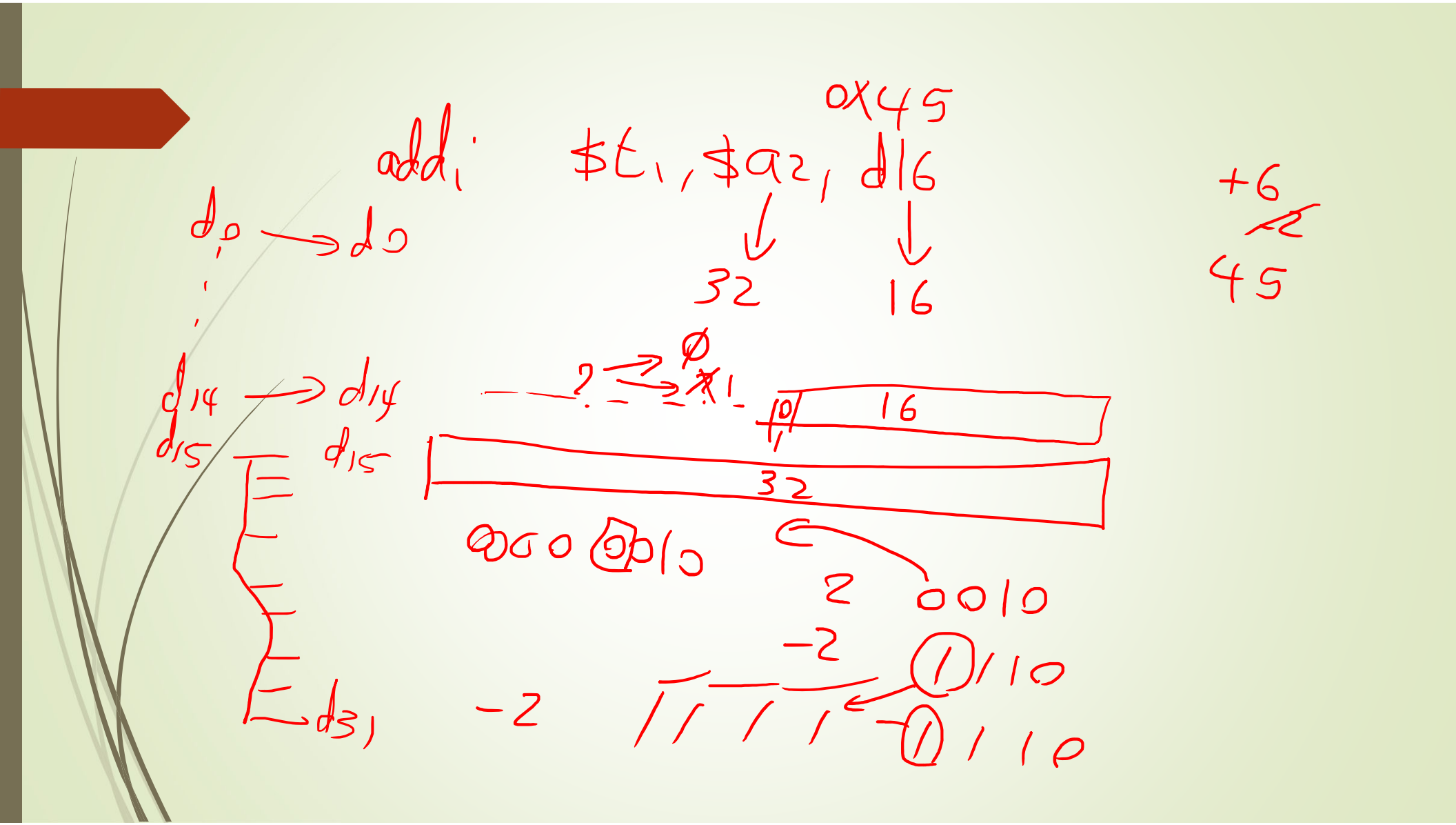
Correct

The mux's top input comes from the data memory. During a load, the data read from data memory gets loaded into a register in the register file.

5) The mux at the upper right either passes PC + 1 (the normal case), or passes a target address from the instruction (for jumps or branches).

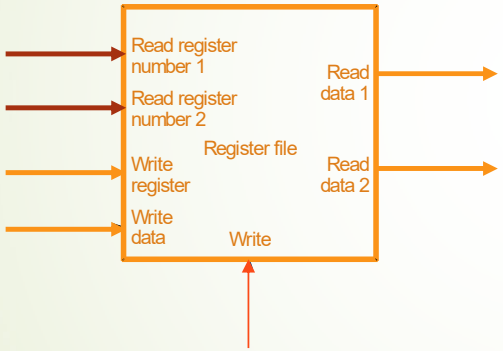
Correct

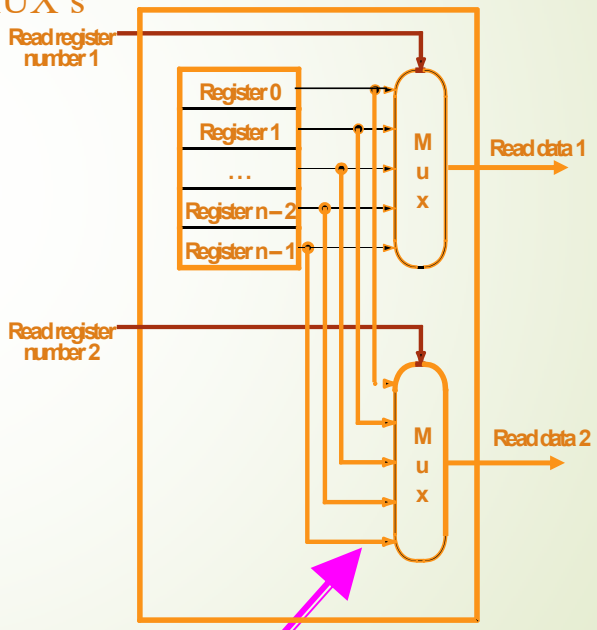
The normal case of fetching the next instruction memory requires PC + 4, not PC + 1, since each word is 4 bytes.


addi $t1, $a2, d16
0x45
+6
45
32
16
$d_0 \rightarrow d_0$
$d_{14} \rightarrow d_{14}$
d_{15} d_{15}
d_{31}
16
32
0000 0010
2 0010
-2 1110
-2 1111 1110

Register File

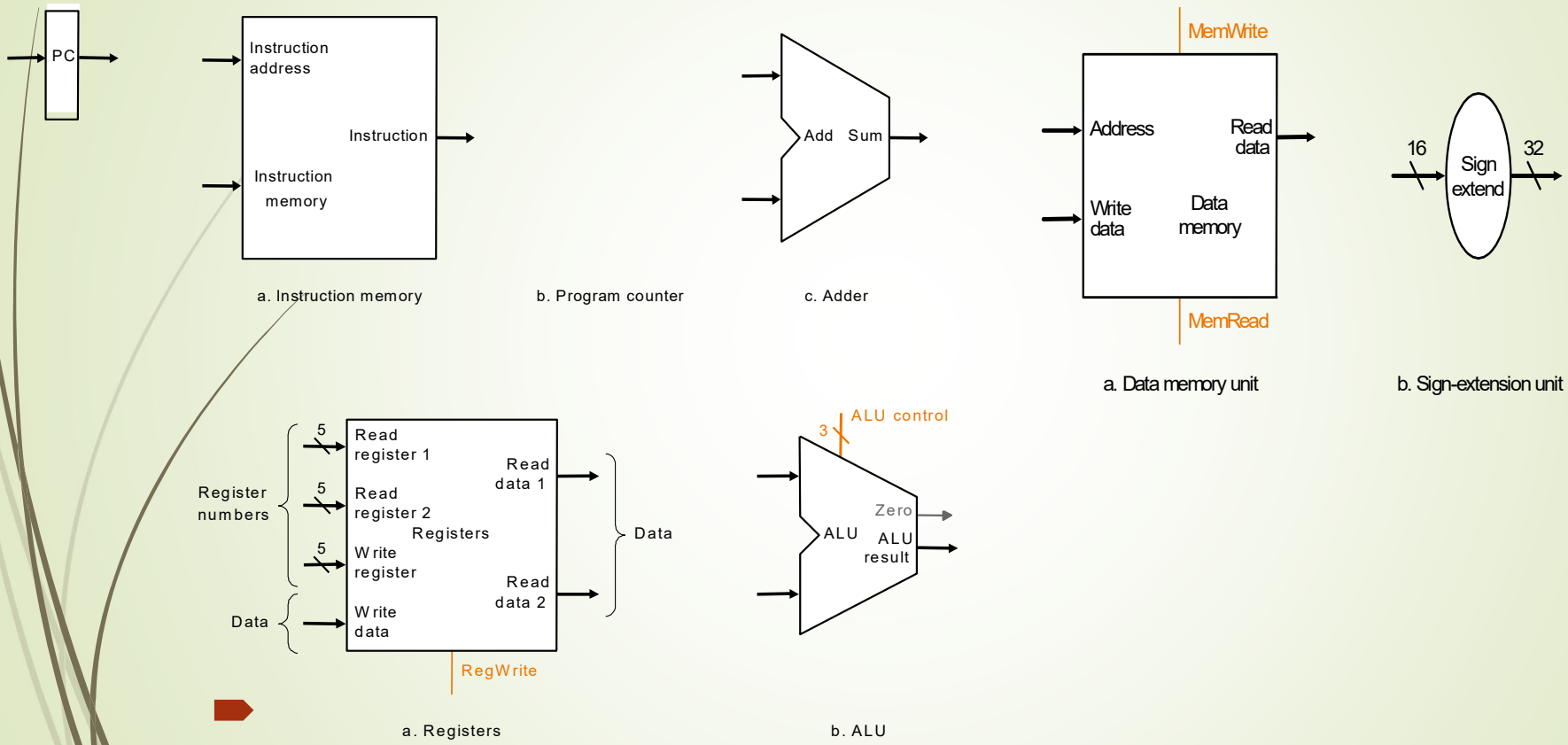
- Read operation using D flip-flops and MUX's

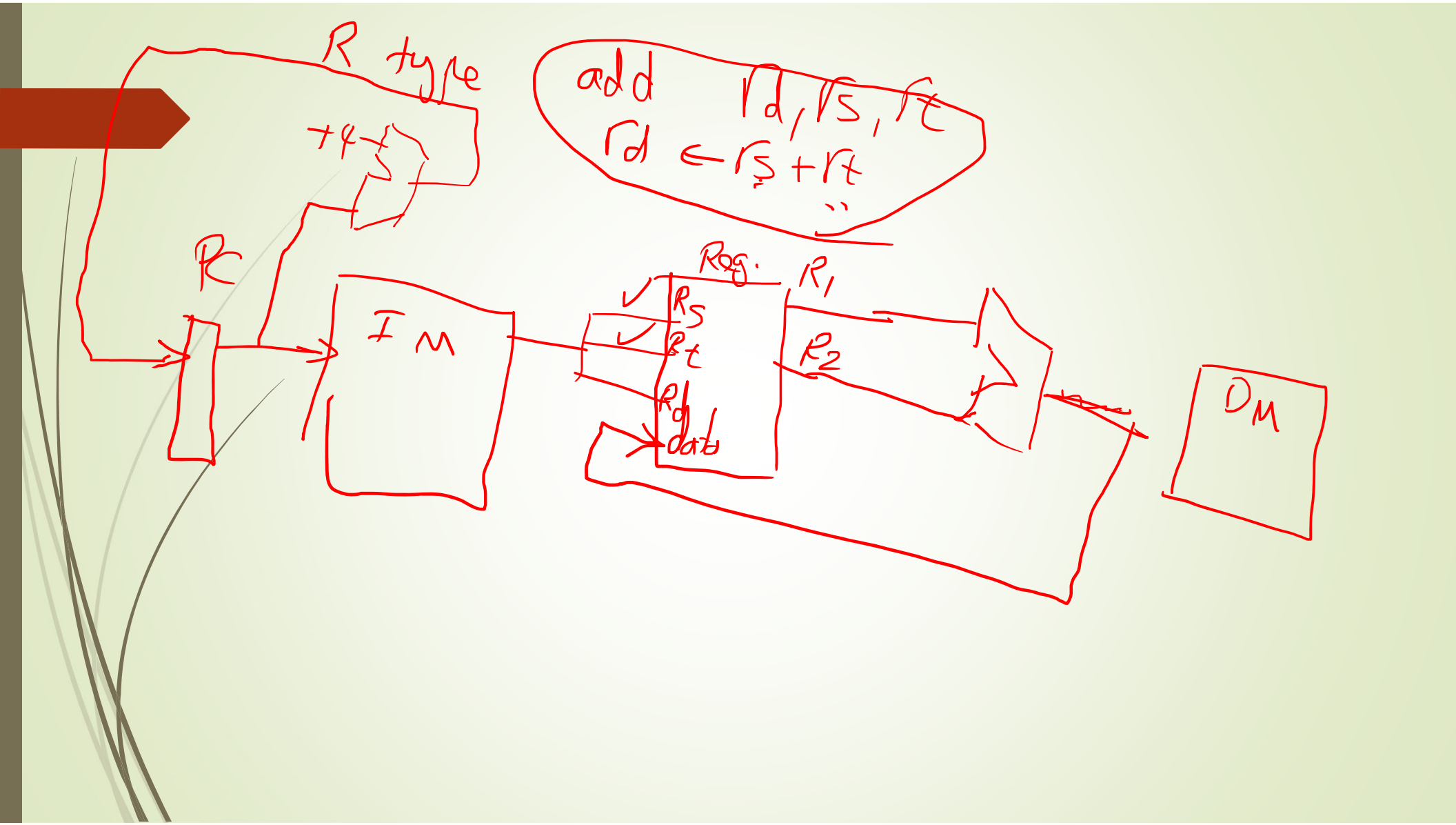




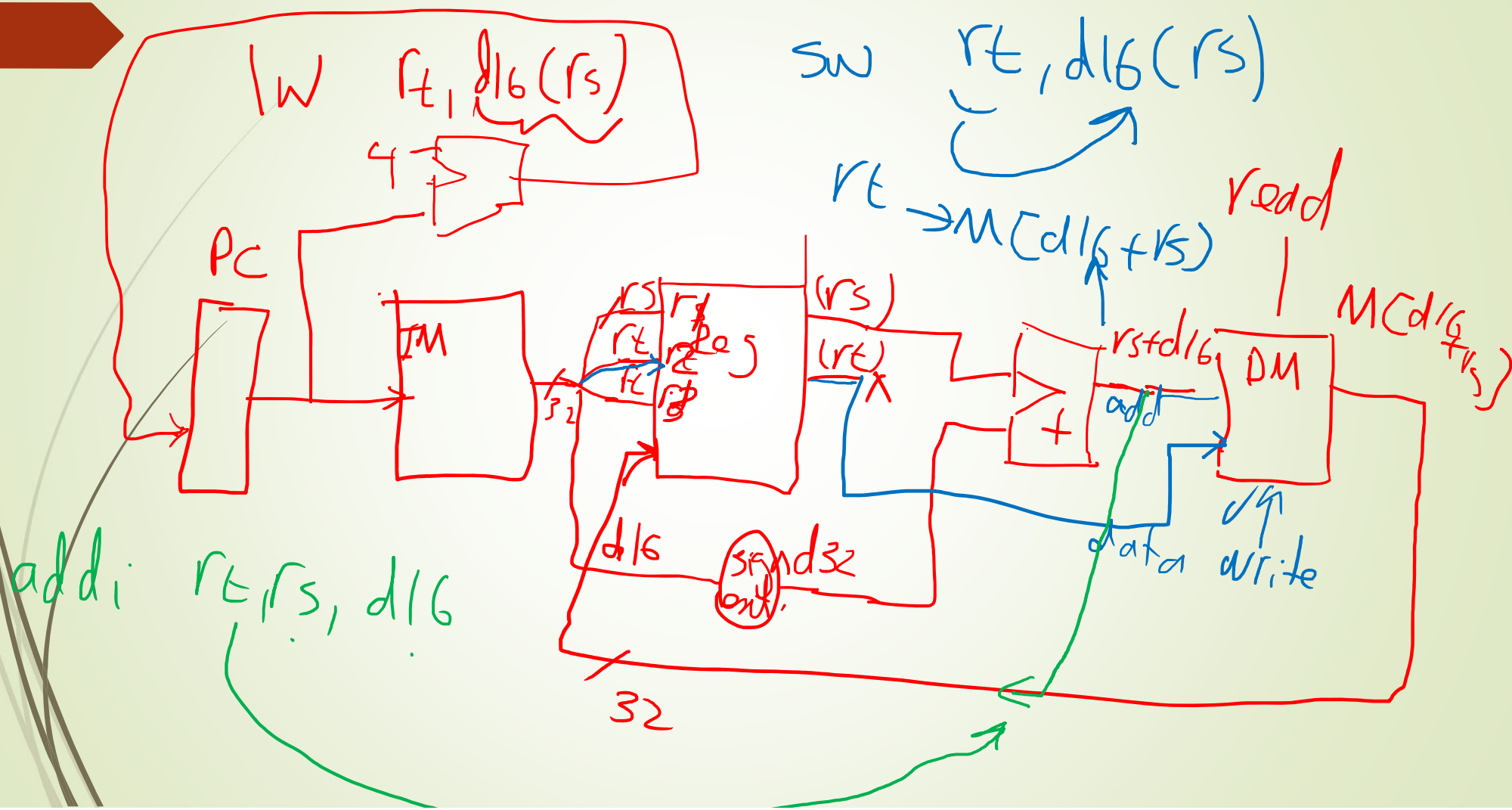
What is the function of "Mux" above?

Building the Datapath Of R



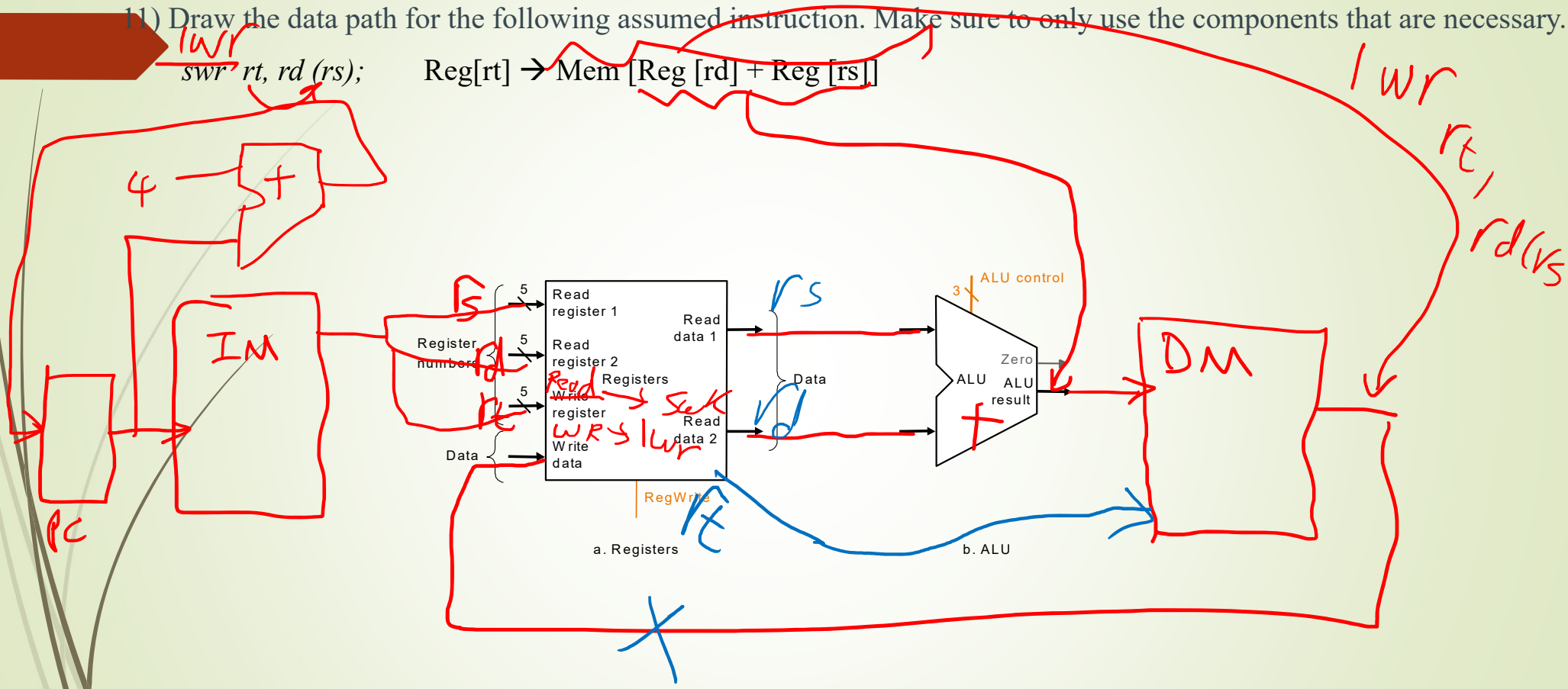

R type
add rd, rs, rt
rd ← rs + rt
PC
I M
Reg.
R1
R2
rs
rt
rd
data
D M

10) Daw the data path for only *lw rt, d16 (rs)*. Make sure to only use the components that are necessary.



11) Draw the data path for the following assumed instruction. Make sure to only use the components that are necessary.

swr rt, rd (rs); Reg[rt] → Mem [Reg [rd] + Reg [rs]]



12) Draw the data path for the following instruction set:

- *lw rt, d16(rs),*
- *sw rt, d16(rs),*
- *R-type,*
- *bne rs,rt, d16,*
- *swr rt, rd(rs)*