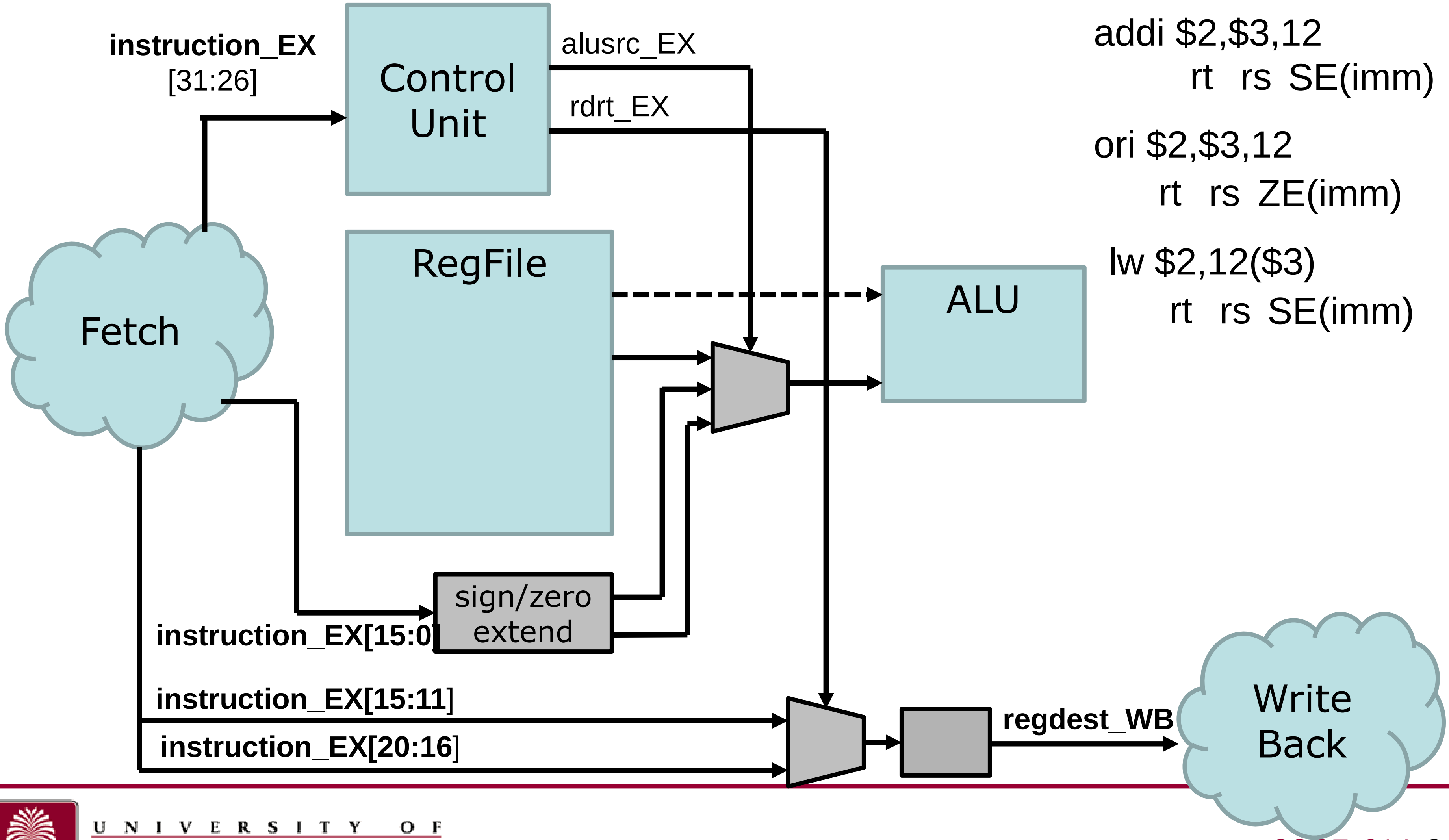
CSCE 611: Execution and Write Back Stages: I-Type ARITH and LOAD/STORE Instructions

Instructor: Jason D. Bakos

Objectives

- Add support for the following instructions:
- **`addi, addiu, andi, ori, xori, slti`**
  - Basically the same as non-immediate versions except:
    - Second input to ALU is sign-extended (for **add** and **slt**) or zero-extended (for **and**, **or**, **xor**) immediate field
    - Destination register is rt (not rd)
- **`lw`**
  - Works exactly like addi, except: (1) combinational ALU output goes out as data memory address, (2) write back data from memory
- **`sw`**
  - Works exactly like addi, except: (1) combinational ALU output goes out as data memory address, (2) contents of rt register to data memory, (3) assert memwrite
- **`lui`**
  - Works like a sll instruction with shamt=16 and uses immediate as input
  - **NOTE: NOT A MEMORY LOAD INSTRUCTION**

EX Stage: New Signals for I-Type ARITH and L/S



addi $2,$3,12
rt rs SE(imm)

ori $2,$3,12
rt rs ZE(imm)

lw $2,12($3)
rt rs SE(imm)

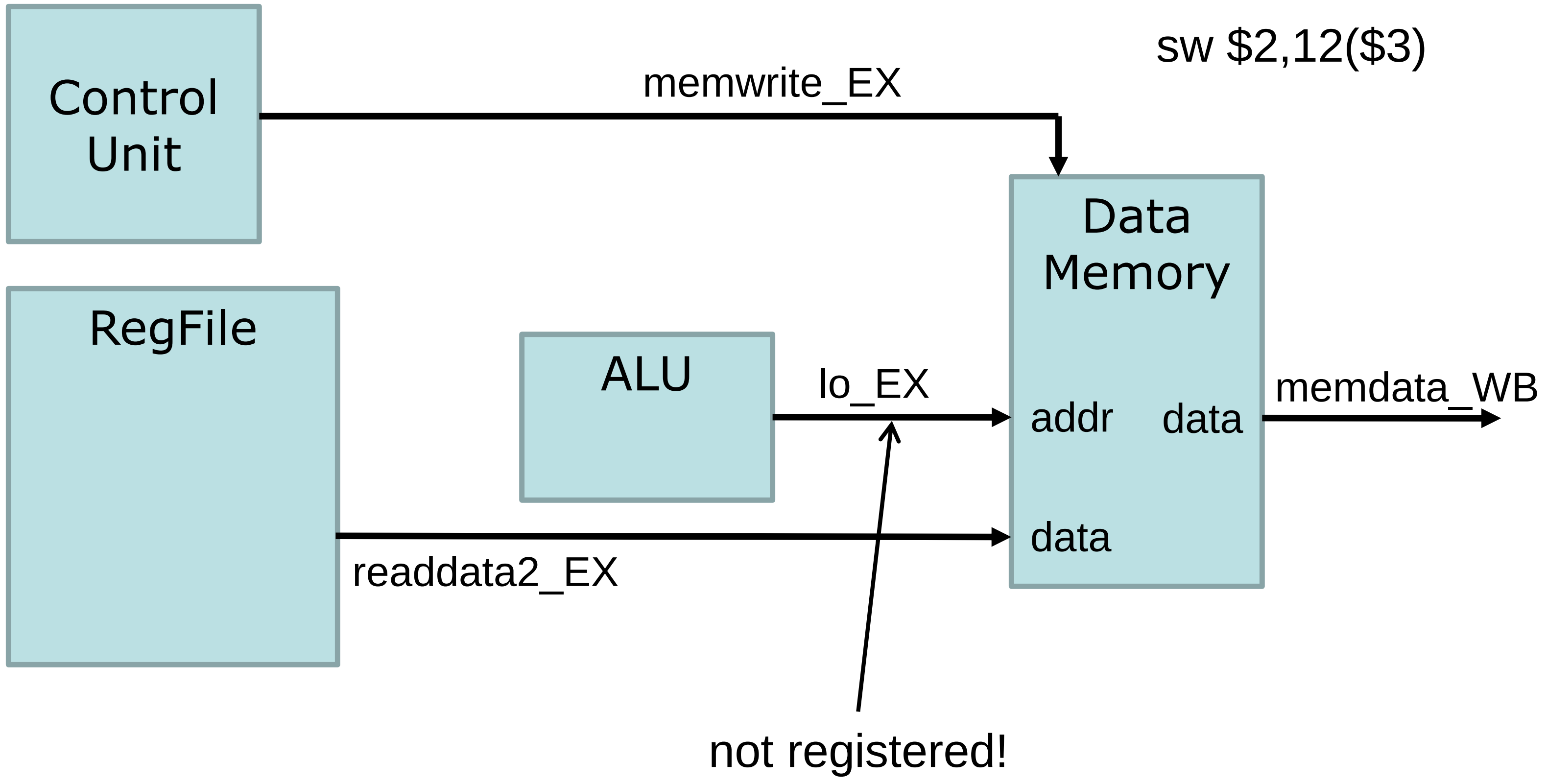
Data Memory

- Need a single port, read/write memory to hold data
- Use a 1024x32 synchronous RAM
- Inputs:
  - [9:0] addr, [31:0] datain, we
- Outputs:
  - [31:0] dataout

EX Stage: New Signals for I-Type ARITH and L/S

lw $2,12($3)

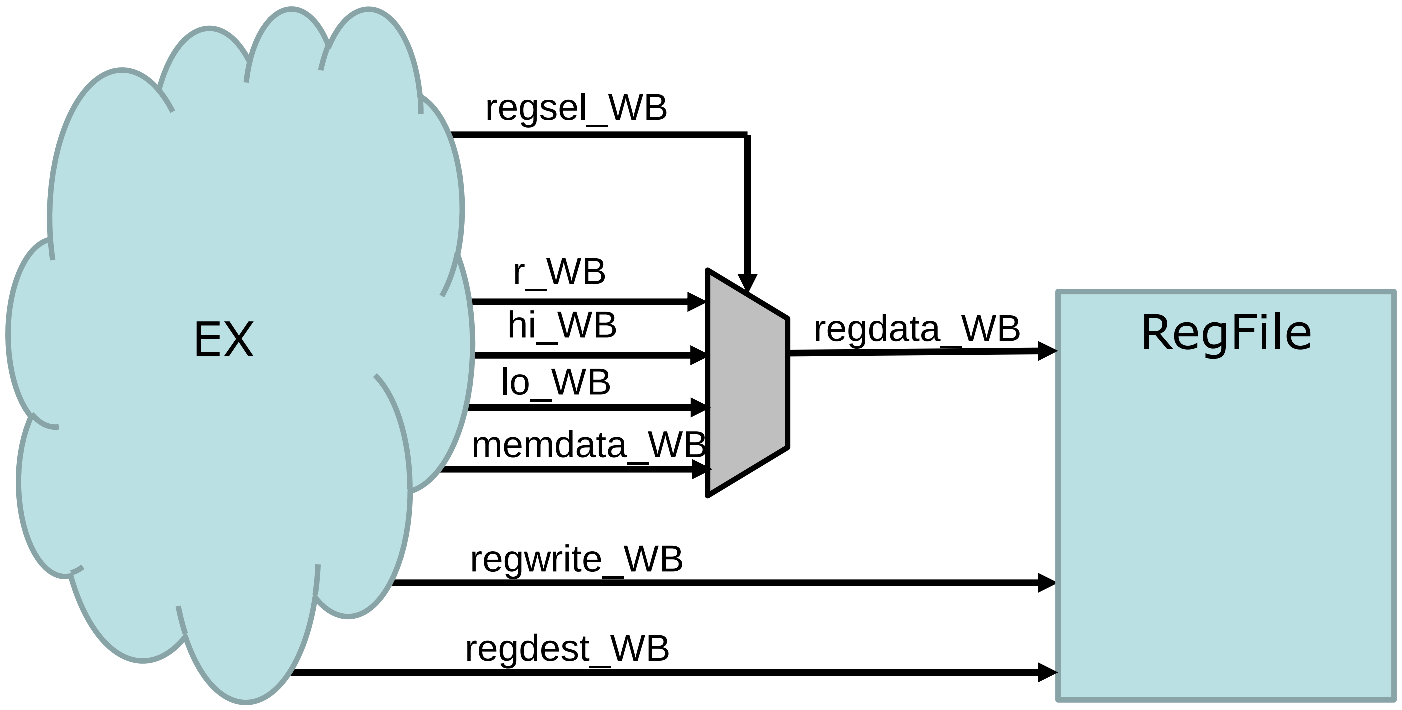
sw $2,12($3)



WB Stage: New Signals for I-Type ARITH and L/S

lw $2,12($3)

sw $2,12($3)



New Control Signals for I-type

- Inputs:
  - **instruction_EX[31:26]:** need to consider non-zero opcodes for all I-type instructions

- Outputs:
  - **memwrite_EX:** set to one for SW
  - **alusrc_EX:** selects between RegFile[rt] and signed- or zero-extended immediate for second ALU input
  - **rdrt_EX:** selects between rd and rt as destination register
  - **regsel_EX:** need to expand to allow for memory data to be written to register file (for LW)

Example Test Program

```
.text
addi $3, $1, 12                             # ($3) <= 23
addiu $4, $2, 13                            # ($4) <= 30
andi $5, $3, 0xf                            # ($5) <= 7
ori $6, $5, 0x10                            # ($6) <= 23
xori $7, $6, 0x1F                           # ($7) <= 8
lui $8,0xBEEF                        # ($8) <= 0xBEEF0000
sw $8,-3($4)                         # address 27 <= 0xBEEF0000
lw $9,4($6)                          # ($9) <= 0xBEEF0000
```

Lab 5: I-type Instructions ARITH and L/S

- Objectives:

  1. Implement MIPS processor that implements instructions:
     `addi, addiu, andi, ori, xori, lui, lw, sw`
  2. Use MARS to generate initialization file
  3. Test with test program
  4. Write another test program
     1. Add up eight values in data memory
     2. Compute arithmetic mean
     3. Display result on 7-segment LEDs
     4. Store result in data memory