# ICS 233
# COMPUTER ARCHITECTURE

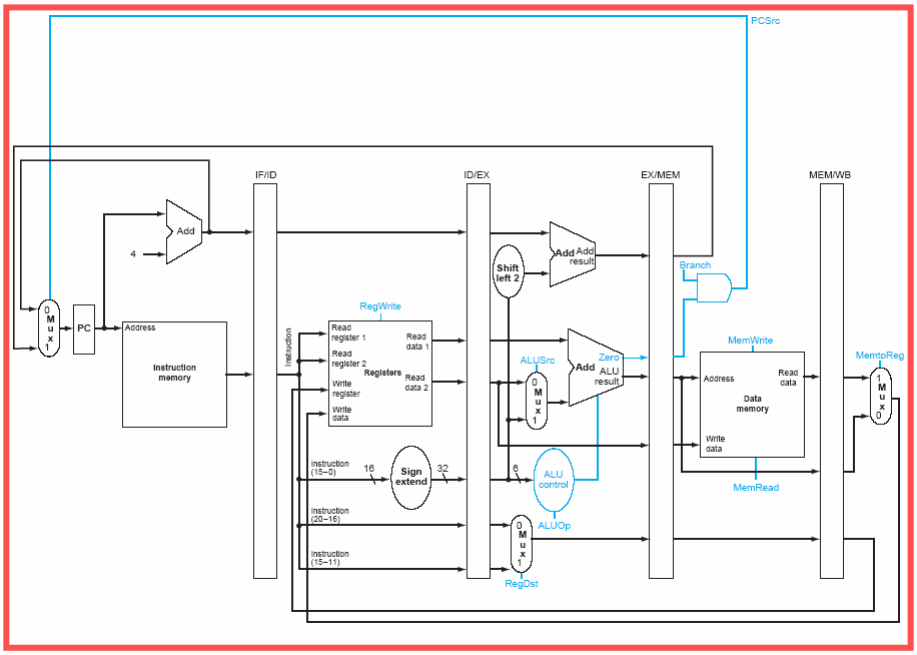
## Pipelined Processor Design

## Enhancing Performance with Pipelining

## Lecture 25



## Pipelined Control



## Pipelined Control

- **To specify control for the pipeline, appropriate control bit values are needed to be set during each pipeline stage**
- **Because each control line is associated with a component active in only a single pipeline stage, the control lines can be divided into five groups according to the pipeline stage :**
  - **Instruction Fetch** :

    The control signals to read instruction memory and to write the PC are always asserted, so there is nothing special to control in this pipeline stage
  - **Instruction Decode/Register file Read** :

    No optional control lines to set
  - **Execution/Address Calculation**

    The signals to be set are **RegDst** (to select the Result register), **ALUOp** (to select ALU operation), and **ALUSrc** (to select either Read data 2 or a sign-extended immediate for the ALU)
  - **Memory Access**

    The control lines set in this stage are **Branch** (set by branch equal instruction), **MemRead** (set by load instruction) and **MemWrite** (set by store instruction).
  - **Write Back**

    The two control lines are **MemtoReg**, which decides between sending the ALU result or the memory value to the register file, and **RegWrite**, which writes the chosen value.



## Building ALU Control Unit

- **Implementing ALU Control**
  - **ALU has four control inputs**
  - **Only six of the possible 16 input combinations are used**

| ALU Control lines | Functions |
|---|---|
| 0000 | AND |
| 0001 | OR |
| 0010 | add |
| 0110 | subtract |
| 0111 | Set less than |
| 1100 | NOR |

- **Depending on the instruction class, the ALU will need to perform one of the five functions**

  **- For load word and store word instructions, use the ALU to compute the memory address by addition**

  **- For the R-type instructions, the ALU needs to perform one of the six actions (AND, OR, subtract, add, set less than, NOR) depending on the value of the 6-bit funct (function) field in the lower order bits of the instruction**

  **- For branch equal, the ALU must perform a subtraction.**

## Building ALU Control Unit

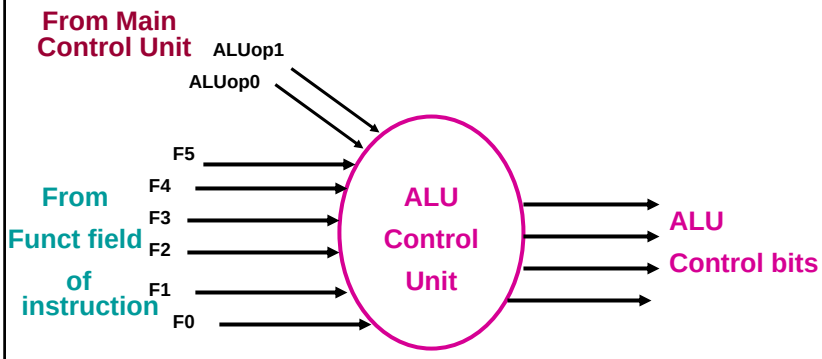
➢ **Implementing ALU Control**

- Generate the 4-bit ALU control input using a small control unit that has as inputs the function field of the instruction and a 2-bit control field called ALUOp.
- ALUOp indicates whether operation to be performed should be
  - add (**00**) for loads and stores
  - subtract (**01**) for beq or
  - determined by the operation encoded in the funct field (**10**)
- The output of the ALU Control unit is a 4-bit signal that directly controls the ALU by generating one of the six 4-bit combinations.



## Building ALU Control Unit

➢ **Implementing ALU Control**



# Building ALU Control Unit

## Implementing ALU Control

**ALU control bits depend on ALUop control bits and Funct field bits**

| Instruction Opcode | ALUop | Instruction operation | Funct field | Desired ALU action | ALU control input |
|---|---|---|---|---|---|
| LW | 00 | Load word | xxxxxx | add | 0010 |
| SW | 00 | Store word | xxxxxx | add | 0010 |
| Branch equal | 01 | Branch equal | xxxxxx | subtract | 0110 |
| R-type | 10 | add | 100000 | add | 0010 |
| R-type | 10 | subtract | 100010 | subtract | 0110 |
| R-type | 10 | AND | 100100 | and | 0000 |
| R-type | 10 | OR | 100101 | or | 0001 |
| R-type | 10 | Set on less than | 101010 | slt | 0111 |

**Truth Table for the four ALU control bits**

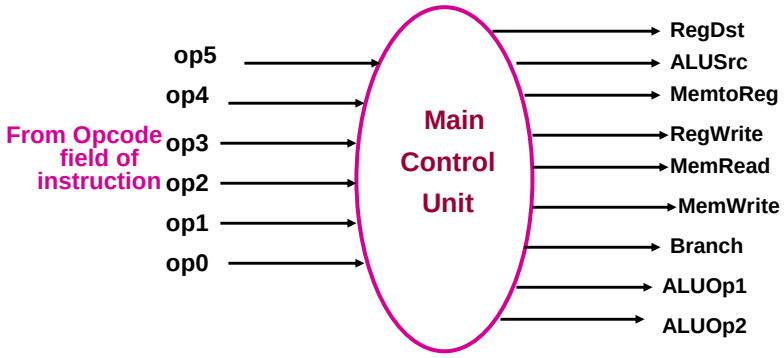
| ALUop | | Funct field | | | | | | Operation |
|---|---|---|---|---|---|---|---|---|
| ALUop1 | ALUop0 | F5 | F4 | F3 | F2 | F1 | F0 | |
| 0 | 0 | x | x | x | x | x | x | 0010 |
| x | 1 | x | x | x | x | x | x | 0110 |
| 1 | x | x | x | 0 | 0 | 0 | 0 | 0010 |
| 1 | x | x | x | 0 | 0 | 1 | 0 | 0110 |
| 1 | x | x | x | 0 | 1 | 0 | 0 | 0000 |
| 1 | x | x | x | 0 | 1 | 0 | 1 | 0001 |
| 1 | x | x | x | 1 | 0 | 1 | 0 | 0111 |

# Main Control Signals

- **To specify control for the datapath, appropriate control bit values are needed to be set during each operation**
- **The control lines associated with components active in an operation can be identified as follows :**

  - **RegDst** - To select the Result register
  - **ALUOp** - To select ALU operation
  - **ALUSrc** - To select either Read data 2 or a sign-extended immediate for the ALU
  - **Branch** - Set by branch equal instruction
  - **MemRead** - Set by load instruction
  - **MemWrite** - Set by store instruction
  - **MemtoReg** - decides between sending the ALU result or the memory value to the register file
  - **RegWrite** - writes the chosen value to the Register file

## Building Main Control Unit






## Building Main Control Unit

### Effect of each of the seven control signals

| Signal name | Effect when deasserted | Effect when asserted |
|---|---|---|
| RegDst | The register destination number for the Write register comes from the rt field (bits 20:16) | The register destination number for the Write register comes from the rt field (bits 15:11) |
| RegWrite | None | The register on the Write register input is written with the value on the Write data input |
| ALUSrc | The second ALU operand comes from the second register file output (Read data 2) | The second ALU operand is the sign-extended lower 16 bits of the instruction |
| PCSrc | The PC is replaced by the output of the adder that computes the value of PC +4 | The PC is replaced by the output of the adder that computes the branch target |
| MemRead | None | Data memory contents designated by the address input are put on the Read data output |
| MemWrite | None | Data memory contents designated by the address input are replaced by the value on the Write data input |
| MemtoReg | The value fed to the register Write data input comes from the ALU | The value fed to the register Write data input comes from the data memory |

# Building Main Control Unit

**Setting of the Control lines Completely determined by the Opcode field bits of the instruction**

| Instruction | RegDst | ALUSrc | Memto Reg | Reg Write | Mem Read | Mem Write | Branch | ALUop1 | ALUop2 |
|---|---|---|---|---|---|---|---|---|---|
| R-format | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| sw | x | 1 | x | 0 | 0 | 1 | 0 | 0 | 0 |
| beq | x | 0 | x | 0 | 0 | 0 | 1 | 0 | 1 |

| Name | Opcode in decimal | Opcode in binary | | | | | |
|---|---|---|---|---|---|---|---|
| | | op5 | op4 | op3 | op2 | op1 | op0 |
| R-format | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| lw | 35 | 1 | 0 | 0 | 0 | 1 | 1 |
| sw | 43 | 1 | 0 | 1 | 0 | 1 | 1 |
| beq | 4 | 0 | 0 | 0 | 1 | 0 | 0 |

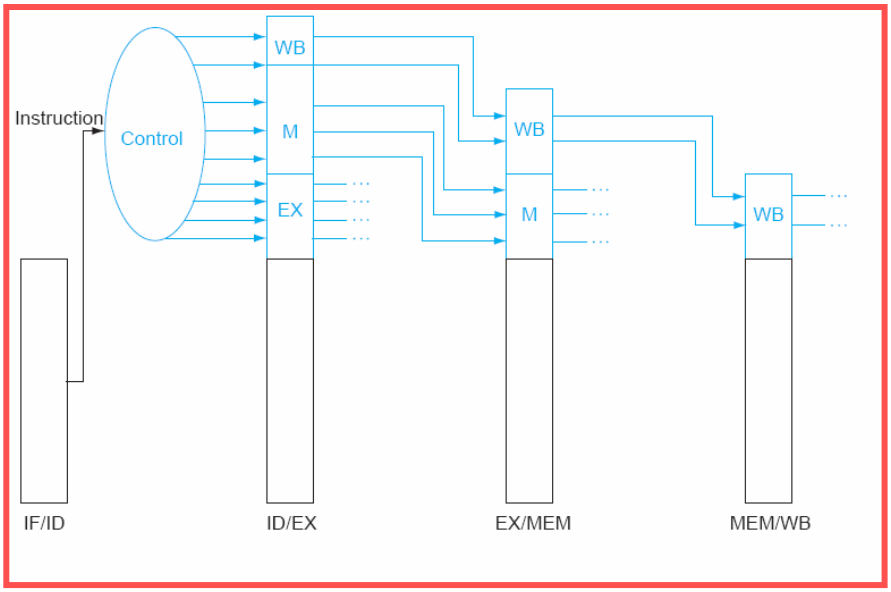


# Building Main Control Unit

**Truth Table for the Main control lines**

| Input or Output | Signal name | R-format | lw | sw | beq |
|---|---|---|---|---|---|
| Inputs | op5 | 0 | 1 | 1 | 0 |
| | op4 | 0 | 0 | 0 | 0 |
| | op3 | 0 | 0 | 1 | 0 |
| | op2 | 0 | 0 | 0 | 1 |
| | op1 | 0 | 1 | 1 | 0 |
| | op0 | 0 | 1 | 1 | 0 |
| Outputs | RegDst | 1 | 0 | x | x |
| | ALUSrc | 0 | 1 | 1 | 0 |
| | MemtoReg | 0 | 1 | x | x |
| | RegWrite | 1 | 1 | 0 | 0 |
| | MemRead | 0 | 1 | 0 | 0 |
| | MemWrite | 0 | 0 | 1 | 0 |
| | Branch | 0 | 0 | 0 | 1 |
| | ALUop1 | 1 | 0 | 0 | 0 |
| | ALUop0 | 0 | 0 | 0 | 1 |

## Pipelined Control






## Pipelined datapath with control signals connected to the control portions of the pipeline registers