# Laboratory 5
# Processor Datapath

## Description of HW Instruction Set Architecture

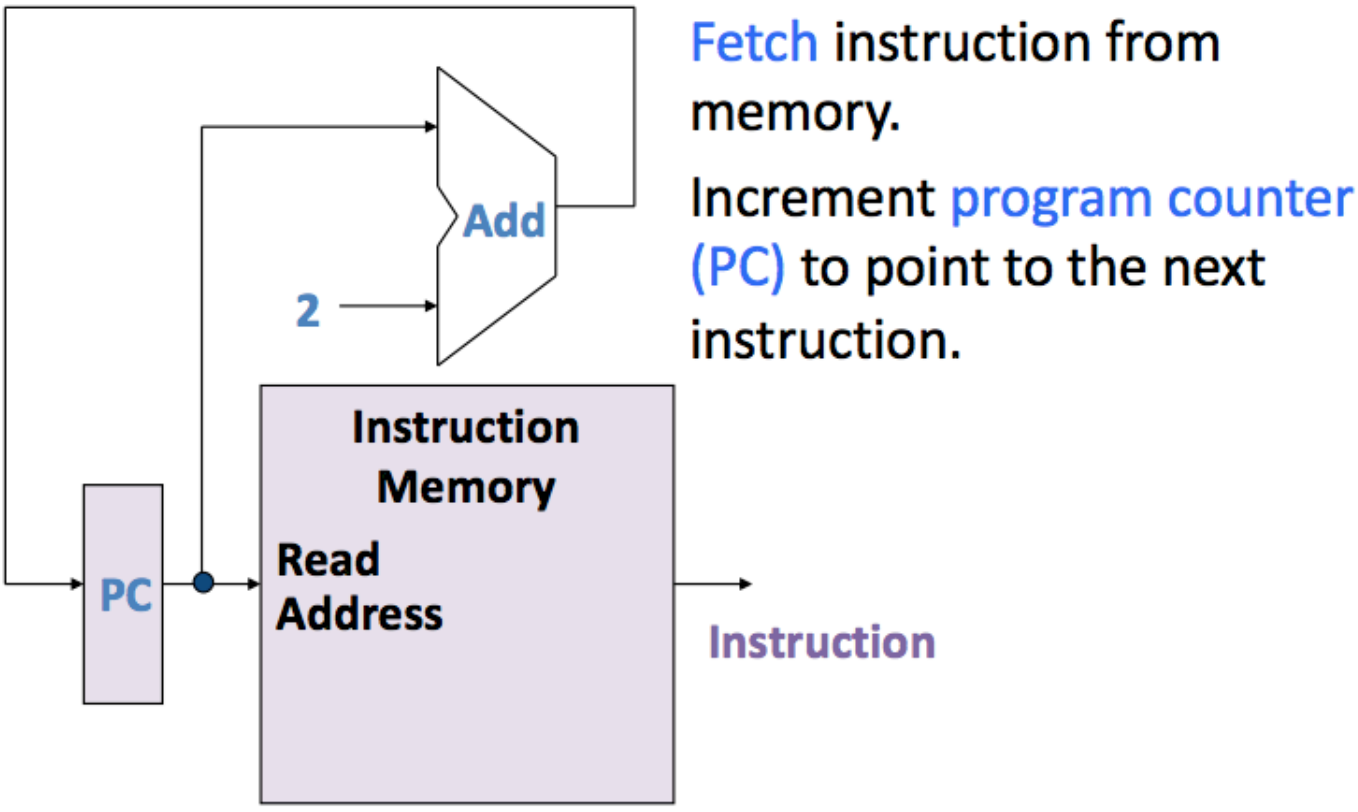
- **16 bit data bus**
- **8 bit address bus**
- **16 registers**
  R0 = 0 (constant)
  R1 =1 (constant)
  R2-R15 general purpose

## HW ISA Instructions

| | | 16-bit Encoding (MSB … LSB) | | | |
|---|---|---|---|---|---|
| **Assembly Syntax** | **Meaning** | **Opcode** | **Rs** | **Rt** | **Rd** |
| ADD R*s*, R*t*, R*d* | R[*d*] ← R[*s*] + R[*t*] | 0010 | *s* | *t* | *d* |
| SUB R*s*, R*t*, R*d* | R[*d*] ← R[*s*] - R[*t*] | 0011 | *s* | *t* | *d* |
| AND R*s*, R*t*, R*d* | R[*d*] ← R[*s*] & R[*t*] | 0100 | *s* | *t* | *d* |
| OR R*s*, R*t*, R*d* | R[*d*] ← R[*s*] \| R[*t*] | 0101 | *s* | *t* | *d* |
| BEQ R*s*, R*t*, *offset* | If R[*s*] == R[*t*] then PC ← PC + *offset**2 | 0111 | *s* | *t* | *offset* |
| JMP *offset* | PC ← *offset**2 | 1000 | *o f* | *f s* | *e t* |

## Fetch Instruction from Memory

- **PC** register holds address of currently executing instruction
- Programs are assumed to start at address 0
- **PC** initialized to 0 by a reset to begin execution
- Next instruction located at current **PC + 2**



# Branch Address

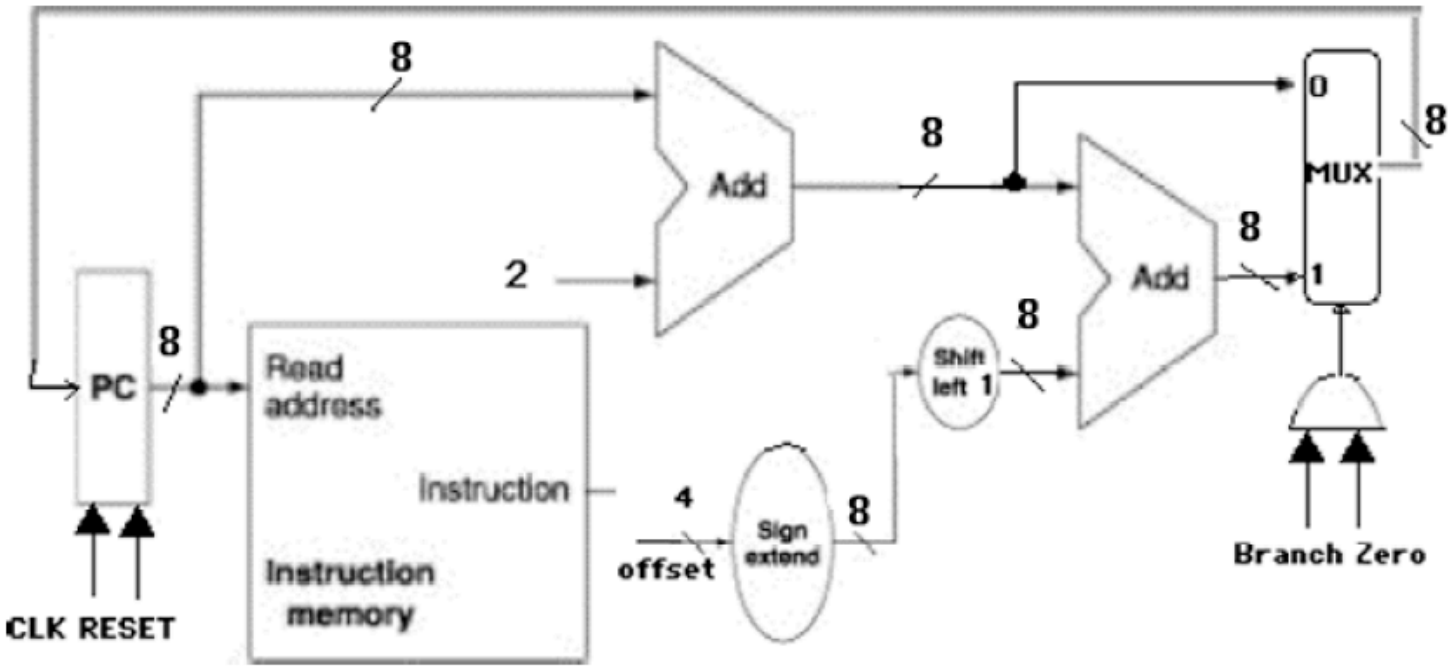
Programs do not always execute in sequential order.

When the **BEQ** instruction is executed, the next instruction to be executed is either:

**PC = PC + 2**

or

**PC = PC + 2 + (2*offset)**




**BEQ Rs,Rt,offset**

- The **offset** = number of instructions away from the next value of the PC to branch to, so must be multiplied by 2.
- Since **offset** is 4 bits, it must be sign-extended to 8 bits to be added to the PC.

A **MUX** selects the next value of the PC. The value of the **Branch** and **Zero** bits are used to determine which is used:

- The **Branch** control line = 1 if a BEQ instruction is being executed.

- The **Zero** bit from the ALU is used to check whether Rs = Rt: it is 1 if Rs – Rt = 0 (meaning they're equal). If **Branch** = 1 and **Zero** = 1, then the next value of the PC will be the branch address ; otherwise, it will simply be **PC + 2**

In the HW computer, there is also an unconditional branch instruction called **JMP** (jump):

**JMP offset**

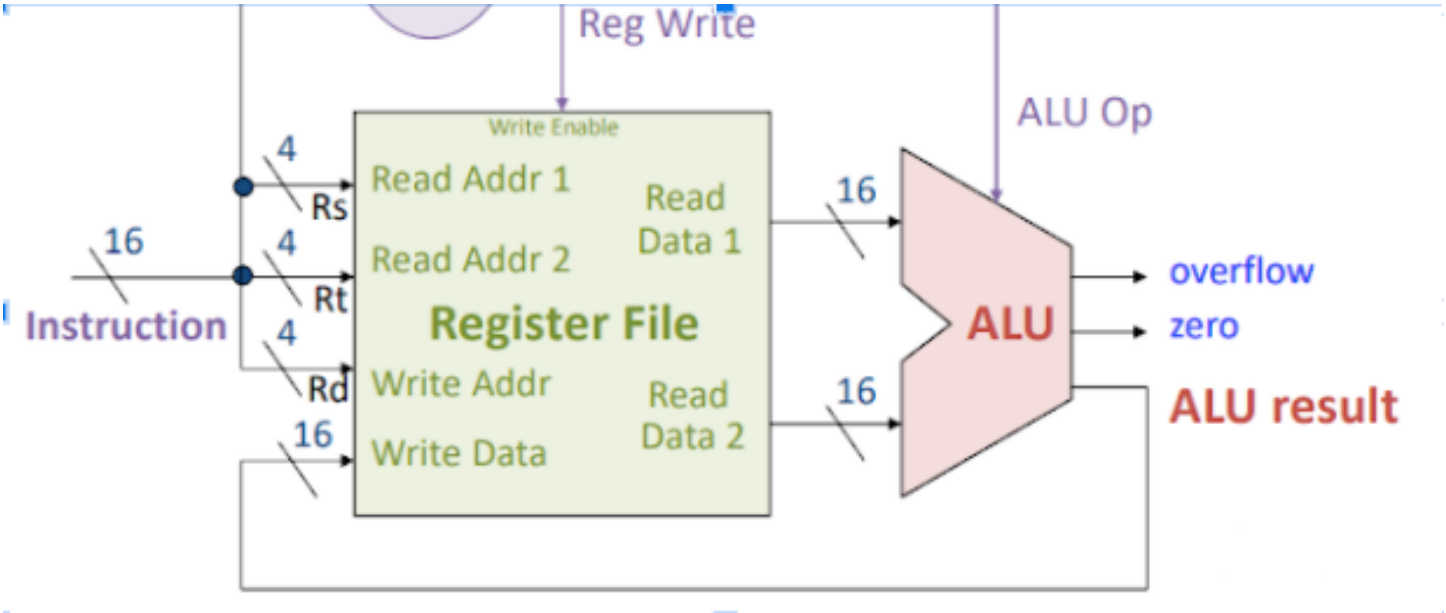
where the *offset* is a 12-bit value which specifies the number of instructions from the beginning of the program to jump to:

**PC = offset * 2**

For example, **JMP 3** sets the **PC** to 0x6, causing the instruction stored at address 0x6 (*i.e.*, the 3rd instruction in the program) to be executed next.

## Datapath

The following diagram describes the basic datapath for executing the arithmetic and logic instructions:




**R-type instructions** ADD,SUB,AND,OR have format: **opcode Rs Rt Rd**

- read **Rs** and **Rt** from register file
- perform an ALU operation on the contents of the registers
- write the result to register **Rd** in register file

ALU can perform 4 possible operations,

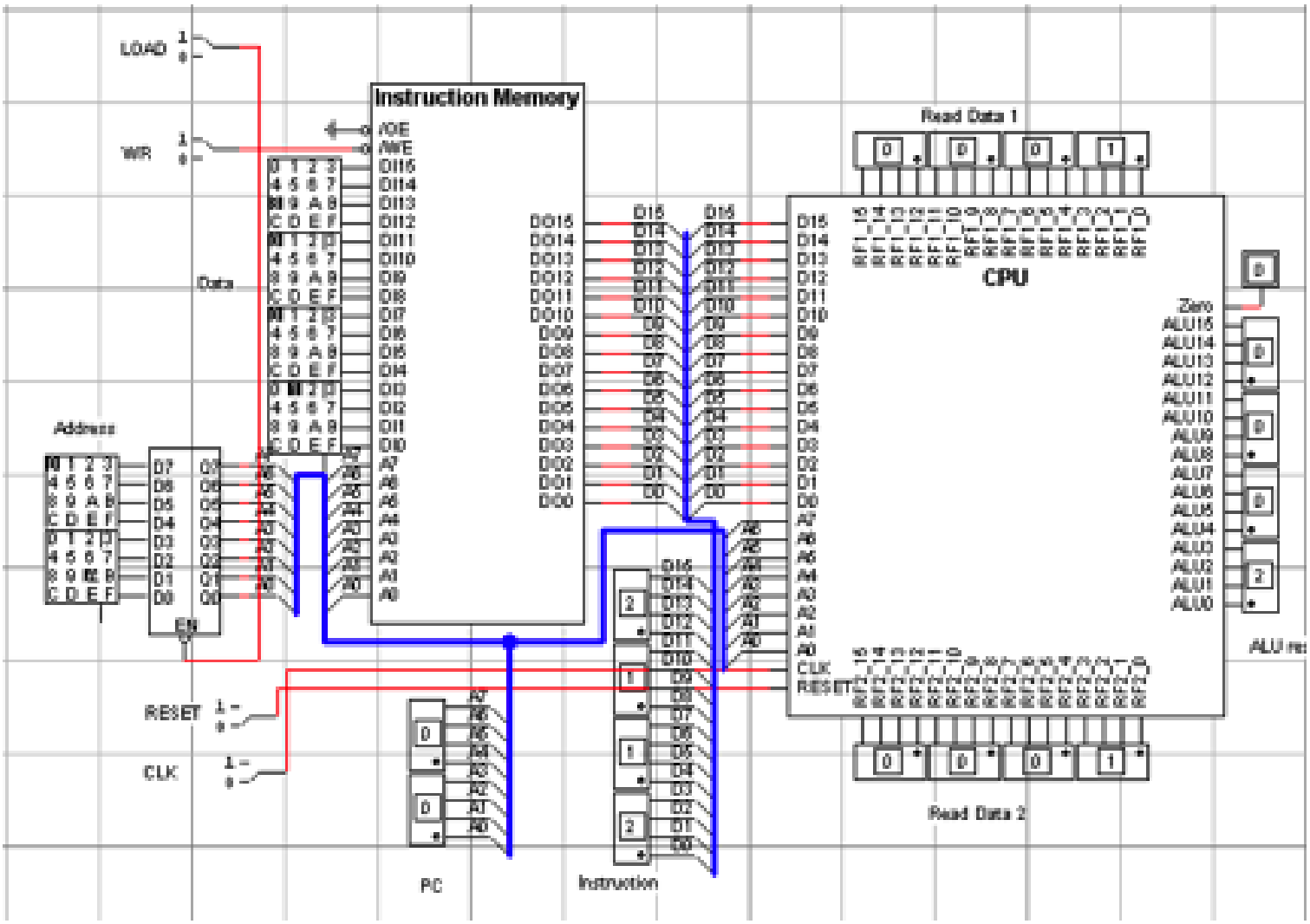
**BEQ** is accomplished by subtraction (which sets the **Zero** bit).

**BEQ** and **JMP** do not change the value of a register.

| **Instruction** | **Opcode** | **ALUop** | **RegWrite** |
|---|---|---|---|
| ADD | 0010 | 0010 | 1 |
| SUB | 0011 | 0110 | 1 |
| AND | 0100 | 0000 | 1 |
| OR | 0101 | 0001 | 1 |
| BEQ | 0111 | 0110 | 0 |
| JMP | 1000 | don't care | 0 |

Can use decoders or simple logic to produce these signals.

## Full Implementation




## Procedure to Load/Execute a New Program

1. Disconnect the address bus of the Instruction Memory from the CPU
2. Set **LOAD** = 0

3. Set **address** and **data** switches for instruction
4. Set **WR** = 0, then back to 1
5. Repeat steps 3 and 4 until all instructions are loaded to memory

6. Set **LOAD** = 1
7. Reconnect address bus to CPU

8. Set **Reset** = 1, then back to 0
9. Set **CLK** = 1, then back to 0, for each instruction.

| Address | Instruction | op | Rs Rt Rd/offset | |
|---|---|---|---|---|
| 0: | | ADD | R1 R1 R2 | # initialize R2 |
| 2: | | ADD | R2 R1 R3 | # initialize R4 |
| 4: | | AND | R2, R2, R4 | # copy R2 to R4 |
| 6: | | AND | R3, R3, R5 | # copy R3 to R5 |
| 8: (LOOP) | | BEQ | R5, R0, 3 | # if R5 is 0, branch to end |
| A: | | SUB | R5, R1, R5 | # decrement R5 |
| C: | | ADD | R4, R4, R4 | # update result in R4 |
| E: | | JMP | 4 | # repeat loop |
| 10: (END) | | AND | R4,R4,R4 | #program will end here with value of R4 |
| | | | | # displayed at outputs of ALU |