**University of Western Ontario, Computer Science Department**
**CS3350B, Computer Organization**

**Assignment 4** **Due: April 7, 2023**

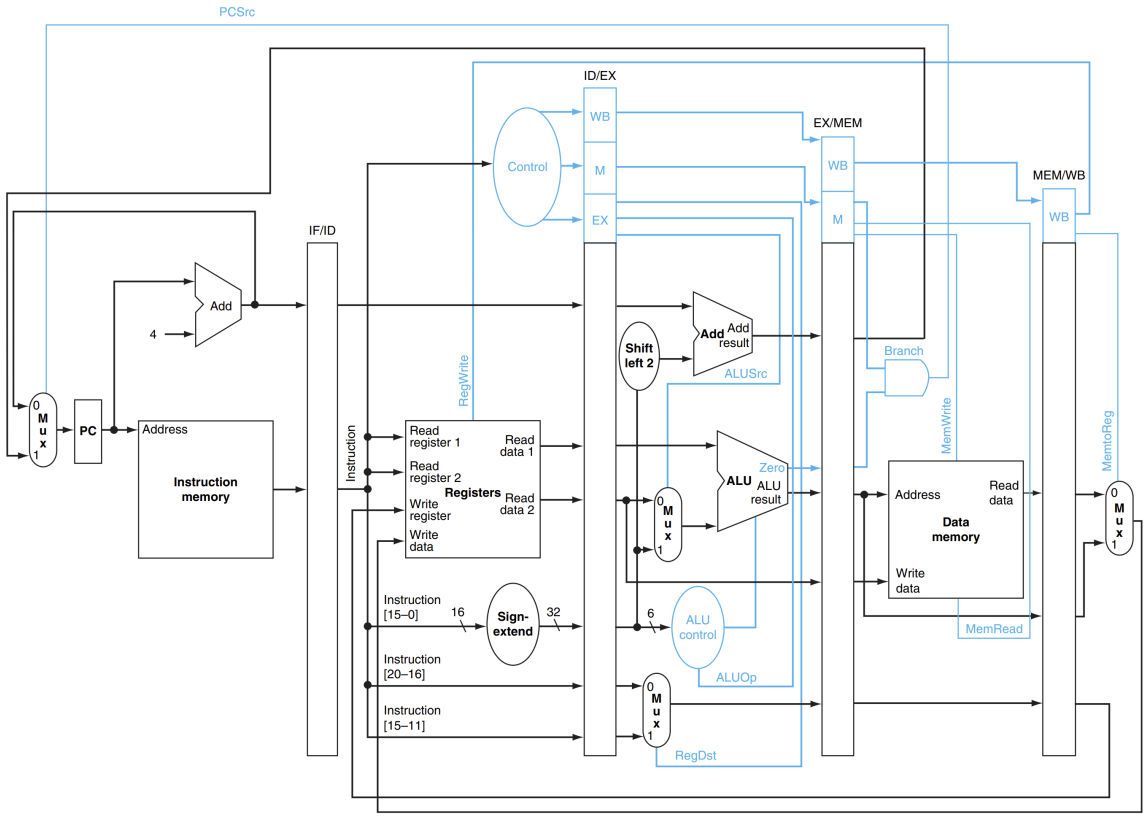


Figure 1: MIPS Datapath with Interstage Registers

**Exercise.** Consider the multi-cycle MIPS datapath presented in Figure 1, it shows 4 inter-stage registers: IF/ID, ID/EX, EX/MEM, and MEM/WB. Consider also the control signals presented in the diagram in blue. Assume the ALUOp control signal is 3 bits. Ignore control signals not shown (i.e. the ones controlling forwarding). Determine the minimum size, in bits, of each of the four inter-stage registers. For part marks, ensure to include workings and not just the total values.

IF/ID: **total 64**
ID/EX: **total 148**
EX/MEM: **total 107**
MEM/WB: **total 71**

**Exercise.**

```
        add $t0, $0, $0     # $t0 = 0, which corresponds to i in C code
loop:   lw $s1, 0($s4)      # assume $s4 stores the base address of array b
        add $s0, $s1, $t0   # $s0 gets b[i] + i
        sw $s0, 0($s2)      # assume $s2 stores the base address of array a
        addi $t0, $t0, 1    # ++i
        addi $s2, $s2, 4    # get address of a[i+1]
        addi $s4, $s4, 4    # get address of b[i+1]
        slt $t2, $t0, $s5   # assume that $s5 holds n
        bne $t2, $0, loop   # if $t2 == 1, go to loop
```

With Loop unrolling applied first:

**(a)**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw | IF | ID | EX | ME | WB | | | | | | | | | | | | | | | | | |
| lw | | IF | ID | EX | ME | WB | | | | | | | | | | | | | | | | |
| nop | | | - | - | - | - | - | | | | | | | | | | | | | | | |
| nop | | | | - | - | - | - | - | | | | | | | | | | | | | | |
| sw | | | | | IF | ID | EX | ME | WB | | | | | | | | | | | | | |
| addi | | | | | | IF | ID | EX | ME | WB | | | | | | | | | | | | |
| nop | | | | | | | - | - | - | - | - | | | | | | | | | | | |
| nop | | | | | | | | - | - | - | - | - | | | | | | | | | | |
| sw | | | | | | | | | IF | ID | EX | ME | WB | | | | | | | | | |
| addi | | | | | | | | | | IF | ID | EX | ME | WB | | | | | | | | |
| addi | | | | | | | | | | | IF | ID | EX | ME | WB | | | | | | | |
| addi | | | | | | | | | | | | IF | ID | EX | ME | WB | | | | | | |
| nop | | | | | | | | | | | | | - | - | - | - | - | | | | | |
| nop | | | | | | | | | | | | | | - | - | - | - | - | | | | |
| slt | | | | | | | | | | | | | | | IF | ID | EX | ME | WB | | | |
| nop | | | | | | | | | | | | | | | | - | - | - | - | - | | |
| nop | | | | | | | | | | | | | | | | | - | - | - | - | - | |
| bne | | | | | | | | | | | | | | | | | | IF | ID | EX | ME | WB |
| | | | | | | | | | | | | | | | | | | | CPI = | 22/10 = 2.2 | | |

**(b)**

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lw | IF | ID | EX | ME | WB | | | | | | | | | |
| lw | | IF | ID | EX | ME | WB | | | | | | | | |
| sw | | | IF | ID | EX | ME | WB | | | | | | | |
| addi | | | | IF | ID | EX | ME | WB | | | | | | |
| sw | | | | | IF | ID | EX | ME | WB | | | | | |
| addi | | | | | | IF | ID | EX | ME | WB | | | | |
| addi | | | | | | | IF | ID | EX | ME | WB | | | |
| addi | | | | | | | | IF | ID | EX | ME | WB | | |
| slt | | | | | | | | | IF | ID | EX | ME | WB | |
| bne | | | | | | | | | | IF | ID | EX | ME | WB |
| | | | | | | | | | | | | CPI = 14/10 = 1.4 | | |

MEM-MEM

MEM-MEM

ALU-ALU (purple is destination of red forwarding, and source of blue forwading)

ALU-ALU

**(d)**

| | ALU or branch | Data transfer | CC |
|---|---|---|---|
| `loop:` | addi $s4 8 | lw $s1 0($s4) | 1 |
| | add $s0 $s1 $t0 | lw $s3 -4($s4) | 2 |
| | addi $t0 $t0 1 | sw $s0 0($s2) | 3 |
| | add $s0 $s3 $t0 | | 4 |
| | addi $t0 $t0 1 | sw $s0 4($s2) | 5 |
| | addi $s2 $s2 8 | | 6 |
| | slt $t2 $t0 $s5 | | 7 |
| | bne $t2 $0 loop | | 8 |

**Exercise.** Consider a multi-core processor with 2 cores, named P1 and P2. Each core has a dedicated cache with the following characteristics:

- 2-way set associative and a 16-byte capacity;
- is initially empty;
- follows the MESI snooping protocol;
- follows write-back and write-allocate protocols; and
- follows a pseudo-LRU replacement policy where
  - (*i*) empty cache lines in a set are filled first, then,
  - (*i*) if there are any invalid cache lines in a set replace them, then,
  - (*ii*) if no invalid cache lines are present, follows a typical LRU replacement policy.

Given the following list of serialized memory byte address accesses by the cores, determine:

(a) whether each access results in a cache hit, cold miss, conflict miss, capacity miss, true share miss, or false share miss;

(b) the data stored in each cache after all addresses in the list have been accessed; and

(c) the MESI state of each cache block after all addresses in the list have been accessed.

| Time | Memory Access | Hit/Miss Type | Time | Memory Access | Hit/Miss Type |
|---|---|---|---|---|---|
| 1 | P1 Reads 5 | cold | 11 | P2 Writes 9 | hit |
| 2 | P2 Writes 8 | cold | 12 | P2 Writes 10 | cold |
| 3 | P1 Reads 9 | cold | 13 | P2 Reads 2 | cold |
| 4 | P1 Writes 14 | cold | 14 | P1 Writes 7 | cold |
| 5 | P1 Reads 3 | cold | 15 | P1 Reads 8 | false |
| 6 | P1 Writes 12 | cold | 16 | P1 Reads 4 | conflict |
| 7 | P2 Reads 6 | cold | 17 | P2 Reads 12 | cold |
| 8 | P2 Reads 17 | cold | 18 | P2 Reads 7 | true |
| 9 | P1 Reads 20 | cold | 19 | P1 Writes 2 | hit |
| 10 | P2 Reads 4 | cold | 20 | P1 Reads 11 | cold |

P1 Cache

| Set | Cache Block Data | | State |
|---|---|---|---|
| 0 | 8 | 9 | S |
| | | | |
| 1 | 2 | 3 | M |
| | 10 | 11 | S |
| 2 | 20 | 21 | E |
| | 4 | 5 | S |
| 3 | 14 | 15 | M |
| | 6 | 7 | S |

P2 Cache

| Set | Cache Block Data | | State |
|---|---|---|---|
| 0 | 8 | 9 | S |
| | 16 | 17 | E |
| 1 | 10 | 11 | S |
| | 2 | 3 | I |
| 2 | 4 | 5 | S |
| | 12 | 13 | E |
| 3 | 6 | 7 | S |
| | | | |