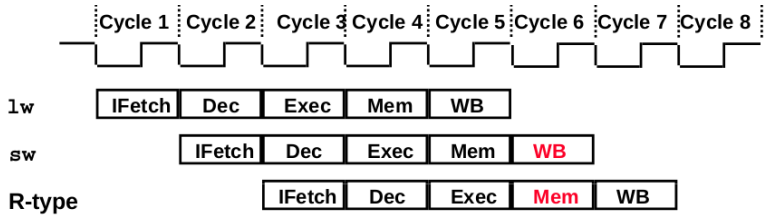
CS3350B Computer Architecture **Quiz 4** **April 2, 2015**

Student ID number:
Student Last Name:

Exercise 1. Consider the sequence of three MIPS instructions being executed by a pipelined processor as depicted on the figure below. What is the CPI (Clock cycles per instruction) of this sequence?

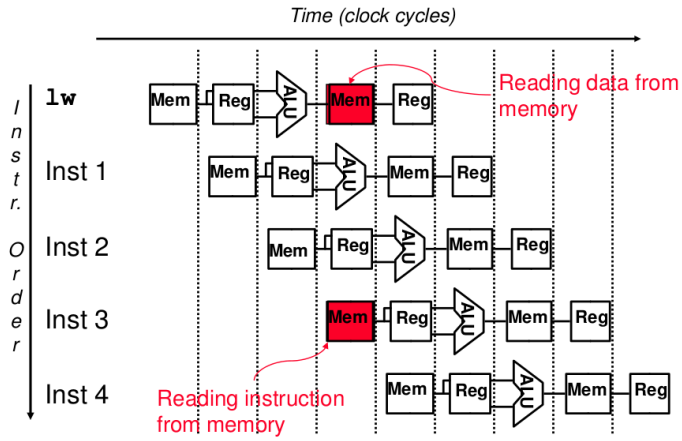


3 instructions into 7 cycles, so CPI is 7/3. (Slide 14 in the PDF version of 6.1)

Exercise 2. What are the hardware implementation techniques (seen in class) that allow a process, on certain instruction sequence, to achieve a CPI (Clock cycles per instruction) less than 1. Briefly detail your answer.

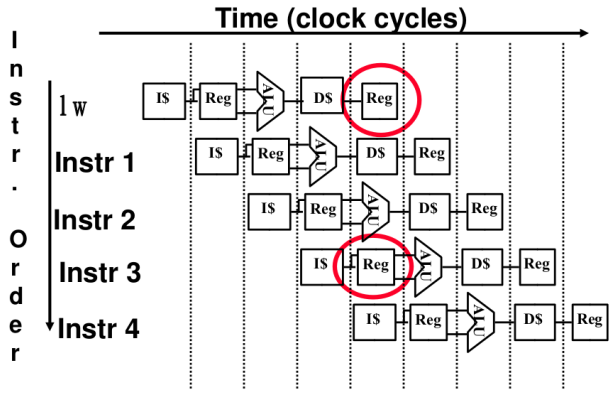
- Pipelined processor with multiple issues
- vectorized instructions, like SSE ones.

Exercise 3. What is the type of hazard depicted on the figure below? How is it resolved at the hardware level?



Slide 4 in the PDF version of 6.2

Exercise 4. What is the type of hazard depicted on the figure below? How is it resolved at the hardware level?



Slides 6 and 7 in the PDF version of 6.2

Exercise 5. What is the type of hazard encountered when executing the sequence of MIPS instructions of the figure below? on a pipelined processor (with a single issue and not supporting instruction re-ordering)? How is it resolved in a a way to minimize the CPI?

```
add $t0, $t1, $t2
sub $t4, $t0, $t3
and $t5, $t0, $t6
or  $t7, $t0, $t8
xor $t9, $t0, $t10
```

Slides 8 to 11 in the PDF version of 6.2

Exercise 6. What is the type of hazard encountered when executing the sequence of MIPS instruction listed below, on a pipelined processor (with a single issue, as on the figure of Exercise 1. Propose a re-ordering of this instruction sequence that preserves semantics and reduces the number of necessary clock cycles to 11.

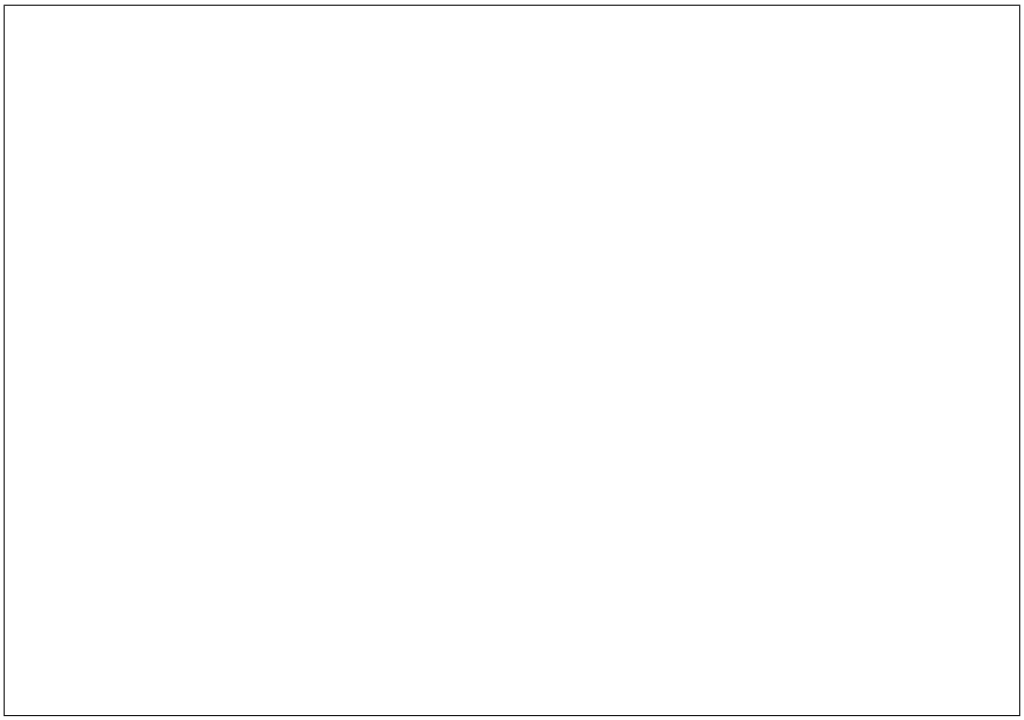
```
lw $t1, 0($t0)
lw $t2, 4($t0)
add $t3, $t1, $t2
sw $t3, 12($t0)
lw $t4, 8($t0)
add $t5, $t1, $t4
sw $t5, 16($t0)
```

See Slide 18 of the PDF version of 6.2.

Exercise 7. Consider the sequence of MIPS instructions of the figure below. Propose a schedule of this sequence on a two-issue pipelined processor with the following constraints:

1. the first issue deals only with ALU or branch instructions,
2. the second issue deals only with data transfer (loads and stored),
3. we wish to avoid pipeline stalls.

```
lp:   lw    $t0,0($s1)   # $t0=array element
      addu  $t0,$t0,$s2  # add scalar in $s2
      sw    $t0,0($s1)   # store result
      addi  $s1,$s1,-4   # decrement pointer
      bne   $s1,$0,lp    # branch if $s1 != 0
```



Exercise 8. Consider again the sequence of MIPS instructions of Exercise 7. Observe that this a MIPS translation of the following C code:

```
for (i=m; i>=0; --i)
   A[i] += n;
```

Assume that `m` is even thus allowing to unroll the loop twice, hence yielding:

```
for (i=m; i>=0; i=i-2)
   A[i] += n;
   A[i-1] += n;
```

What is its MIPS translation of this unrolled loop?

Propose a schedule of this new sequence of MIPS instructions on a two-issue pipelined processor with the same constraints as in Exercise 7:

See Slide 13 on the PDF version of 6.3 (4-way loop unrolling).