Last NAME: RAMOS First Name: ANTHONY

Computer Science
C.Sc. 342/343

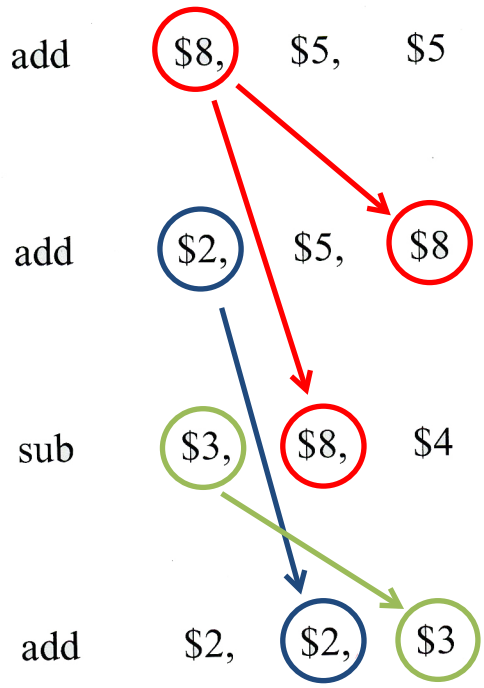
Quiz PIPELINED Processor
CSc or CPE

May 12, 2021

Please write your name on every page.
NO CORRECTIONS ARE ALLOWED !!!!! You may use back page for notes.
Please answer all questions. No computers are allowed.

Question 1: (40 points) Pipelined MIPS processor from the book.

Show or list all of the dependencies in this program. For each dependency, indicate which instructions and register are involved.



You can draw the dependencies using ARROWS, or describe them in words:

Last NAME: RAMOS First Name: ANTHONY

Question 2: Pipelining performance(40 points)

IPG	FET	ROT	EXP	REN	WLD	REG	EXE	DET	WRB
1	2	3	4	5	6	7	8	9	10

One CPU manufacturer has proposed the 10-stage pipeline above for a 500MHz (2ns clock cycle) machine.
Here are the correspondences between this and the MIPS pipeline:

- **Instructions are fetched in the FET stage.**
- **Register reading is performed in the REG stage.**
- **ALU operations and. memory accesses are both done in the EXE stage.**
- **Branches are resolved in the DET stage.**
- **WRB is the writeback stage.**

Q 2.1

How much time is required to execute one million instructions on this processor, assuming there are no dependencies or branches in the code?

2.1.1 Number of Cycles to fill the pipeline = 9 Cycles ????

2.2.2 Number of Cycles to complete 1 million instructions= 1M Cycles ????

2.2.3 Total time in (ns) to compute 1 million instructions is = ?????? 1,000,009 * 2ns = 2,000,018ns

Q 2.2 If a branch is mispredicted, how many instructions would have to be flushed from the pipeline? 3 (From IF,ID, EX Stages)

Q 2.2.1 Branches aren 't resolved until the MEM ?????? stage (please write the stage label).

Q 2.2.2 Number of instructions that have to be flushed from the pipeline is = 3 (From IF,ID, EX Stages) ??????

Q 3 (20 Points) Reordering Code to Avoid Pipeline Stalls

Consider the following code segment in C:

```
A=B+E;
C=B+ F;
```

Here is the generated MIPS code for this segment, assuming all variables are in memory and are addressable using relative addressing mode i.e. using offsets from register $t0:

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

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

Find the hazards in the following code segment explain, and reorder the instructions to avoid any pipeline stalls.

There are several stalling hazards. Both 'add' instructions are dependent on the load (lw) instruction that comes immediately before it. However, if we move the third lw instruction (in green) before the first 'add' instruction, we can bypass the aforementinoed stalls.