#1. (21 points, 3 points each) Circle T for true or F for false.

* Pipelining increases instruction throughput, rather than decreasing instruction latency.  
  T

* A single-cycle datapath allows you to reuse resources such as adders for the same instruction, something not possible with a multi-cycle datapath. 
  F

* It is possible to receive arithmetic overflow exceptions with a typical C program using a typical C compiler, if you use signed instructions. 
  F

* It is possible to receive arithmetic overflow exceptions in an assembled MIPS program if you use the signed MIPS arithmetic instructions. 
  T

* An instruction placed in the branch delay slot is only executed if the branch is not taken. 
  F

* Tomasulo’s algorithm performs a dynamic form of register renaming. 
  T

* "Branch target buffer" is just another word for "branch predictor". 
  F
#2 through #5 are multiple choice, circle all answers that apply (i.e., your answer could be more than one of the choices, or none). As the given sequence of instructions makes its way through a MIPS pipeline as described in the book, and assuming no exceptions occur and all branch delay slots are appropriately filled (i.e., branches require no bubbles that affect IPC), which of these things **might** happen (assume that we are doing branch prediction and any branches may or may not be taken, also assume no dependencies for the instructions you can't see, i.e., the ones that come before and after. Consider each block of code in isolation.):

#2. (5 points).

```
addi $t0, $t1, 5
subu $t2, $t1, $t5
beq $t5, $zero, Label
```

a. A bubble will be inserted by the hazard detection unit that decreases IPC.
b. Forwarding between EX/MEM and ID/EX will occur.
c. Forwarding between MEM/WB and ID/EX will occur.

d. **The pipeline might be flushed.**

#3. (5 points).

```
lw $t5, 24($t4)
subu $t2, $t4, $t0
addi $s0, $t5, -18
sll $s1, $s1, 3
add $t3, $t5, $s2
```

a. A bubble will be inserted by the hazard detection unit that decreases IPC.
b. Forwarding between EX/MEM and ID/EX will occur.
c. **Forwarding between MEM/WB and ID/EX will occur.**
d. The pipeline might be flushed.
#4. (5 points).

```
lw $t2, 24($t6)
subu $t5, $t6, $t2
add $s3, $s3, $s0
sll $s1, $s1, 3
add $t4, $t7, $s4
```

* a. A bubble will be inserted by the hazard detection unit that decreases IPC.
* b. Forwarding between EX/MEM and ID/EX will occur.
* c. *Forwarding between MEM/WB and ID/EX will occur.*
* d. The pipeline might be flushed.

#5. (5 points).

```
lw $t7, 24($t6)
subu $t5, $t6, $t2
add $s3, $s3, $s0
sll $s1, $s1, 3
add $t4, $t7, $s4
```

* a. A bubble will be inserted by the hazard detection unit that decreases IPC.
* b. *Forwarding between EX/MEM and ID/EX will occur.*
* c. Forwarding between MEM/WB and ID/EX will occur.
* d. The pipeline might be flushed.
#6. (10 points) Draw lines to divide the following MIPS code into basic blocks, and then indicate below the total number of basic blocks. Two points for each division line you draw in a correct place and the rest of the points for getting the total right.

```
addi $t0, $t0, 1
sub $t5, $t6, $t7
bne $t0, $t5, Label1
sub $s1, $s1, $s2
sll $s2, $s2, 1
Label1: addi $s3, $s2, -4
li $a0, 6
jal SomeFunction
```

How many basic blocks are there total? ____3____

#7. (10 points) Using the following equation for Ahmdahl’s law, and showing your work, answer the following question.

\[
O.S. = \frac{1}{(1 - f) + \left(\frac{f}{s}\right)}
\]

Suppose that some percentage of your clock cycles for a given suite of benchmarks are spent stalling because the register file is busy. You figure that you can effectively speedup these stall cycles by a factor of 4 by multi-porting the register file. What fraction of cycles must be this type of stall for you to achieve an overall speedup of 4/3?

Plug in \( O.S. = \frac{4}{3} \) and \( s = 4 \), solve for \( f = \frac{1}{3} \).
#8. (10 points, 2 point each.) Write the name of the addressing mode to which each constraint applies.

Limited to whatever 256MB block of memory the PC currently is in, not relative to the PC or any register:

_________Pseudo-direct addressing________________

Limited to 32 registers: _________Register addressing______________

Limited to a immediates between $-2^{15}$ and $(2^{15}-1)$: _Immediate addressing___

Limited to branching to within about a 256 KB range relative to the current program counter plus 4:

_____PC-relative addressing________

Limited to loading or storing a word, halfword or byte that is + or – about a $2^{15}$ offset from an address that is contained in a register:

________Base or displacement addressing____

*#9, (20 points) Multiple choice, 4 points each. Circe all answers that apply. Your answer could be all of them, none of them, or any subset. These will be graded like four 1-point multiple-choice questions.

*Which of the following are structures would you expect to find in a typical superscalar CPU that uses Tomasulo’s algorithm:

a. A reorder buffer.
b. A common data bus.
c. A VLIW issue window.
d. Reservation stations.
*For which of the following might it be possible to measure IPC > 1.0 for some given benchmark:

   a. superscalar processor with issue and commit widths of 4.
   b. A static multiple issue processor with issue and commit widths of 4.
   c. A scalar pipeline without forwarding support.
   d. A scalar pipeline with forwarding support.

*For which of the following might it be possible to measure CPI > 1.0 for some given benchmark:

   a. superscalar processor with issue and commit widths of 4.
   b. A static multiple issue processor with issue and commit widths of 4.
   c. A scalar pipeline without forwarding support.
   d. A scalar pipeline with forwarding support.

*On an Exlicitly Parallel Instruction Computer like the Itanium, what must be true of the instructions between any two stops?:

   a. They must have no data dependencies between them.
   b. There can only be as many of them as the issue width allows (which is why VLIW is better than EPIC).
   c. They cannot be in the reorder buffer at the same time.
   d. None of the instructions can be predicated.

*Which of the following are ways you might deal with either structural or control (but not data) hazards?

   a. Multi-porting the register file.
   b. Branch delay slot.
c. Using a separate adder for calculating the PC.

d. Branch prediction.

#10. (4 points) Be as concise as possible but still answer the question. What is a correlating branch predictor?

A correlating branch predictor is a branch predictor that combines local and global behavior by considering not only the history of the current branch being predicted, but of other branches in the code as well.

#11. (2 points) Why is a WAR hazard known as a anti-dependence or false dependence?

Because only the fact that a register is getting re-used makes it a problem to reorder the instructions, register renaming will completely remove the hazard so that there is no dependency.

#12. (1 point) What type of cache does the book identify as absolutely critical for superscalar processors?

A non-blocking cache.

#13. (1 point) What is the ratio of physical registers to architectural registers in the Pentium 4 as described in Chapter 6 of the book?

128:8

#14. (1 point) In the movie I.Q. starring Walter Mathau as Albert Einstien, Lou Jacobi plays a mathematician friend of Einstein's known for his perfectly round glasses and developing an incompleteness theorem related to Turing undecidability. What was the real-life mathematician's name?

Kurt Gödel