1. (10) Answer each of the following statements as being true or false (circle T or F).

MIPS (million instructions per second) is a good measure of performance.

False

Pipelines increase CPU performance by decreasing instruction latency.

False, *pipelines increase throughput*

A single-cycle datapath typically has a longer clock cycle than a multi-cycle datapath.

True

A single-cycle datapath can re-use resources such as an adder for the same instructions while a multi-cycle datapath cannot.

False, it's the other way around

A branch target buffer and a branch predictor are two names for the same thing.

False

Loop unrolling is typically not used by compilers for static multiple issue microarchitectures because it doesn't help performance for static multiple issue.

False
The Pentium 4's much deeper pipeline (than a typical MIPS processor for example) increases both the clock rate and the IPC.

True

Tomasulo's algorithm uses reservation stations and a reorder buffer.

True

Tomasulo's algorithm does not perform any form of register renaming.

False

A RAW (Read-After-Write) dependency is a false dependency, also sometimes called anti-dependency.

False, WAR hazards are false dependencies

2. (10) You will need the formula for Amdahl's law for problems 2 and 3:

OVERALL SPEEDUP = 1 / ((1 – f) + (f / s))

Where f is the fraction of the execution time that a speedup applies to, and s is the speedup for that fraction only.

What is the maximum overall speedup you can hope to achieve by focusing on an improvement that applies to one fifth (20%) of your execution time? Be sure to show your work by plugging numbers into the equation above.

Plug in f = 0.2, s = infinity, solve for O.S. = 5/4
3. (10) You have two algorithms to choose from for calculating the SVD of a large, sparse matrix. One runs in serial only, it can not be parallelized, and takes 2.0 teraflops to complete a benchmark you are interested in. The other can be parallelized, but only 2/3 (66.6%) of the serial execution time can be parallelized. In total, this algorithm takes 3.0 teraflops in serial execution time. The machine you have available to run your code has two dual-core processors (so 4 CPUs available). Which algorithm should you choose? (Show your work by plugging numbers into the equation above, ignore communication overhead, etc.).

Plug in f = 2/3, s = 4, solve for O.S. = 2 for parallel case, meaning that you have to do 3.0/2 = 1.5 units of time for the 3.0 flops versus 2 units of time for the serial.

4. (20) Answer these five multiple choice questions. Circle all that apply, i.e. you may need to circle multiple answers for any question:

*IPC can be affected by changes in:
  a. The Instruction Set Architecture
  b. The clock rate
  c. The microarchitectural design of the CPU
  d. The accuracy of the branch predictor
*Structures that enable the implementation of precise interrupts in Tomasulo's algorithm include:

a. The Reorder Buffer  
b. The Future File  
c. The Sign Extension Unit  
d. A multi-ported memory

*Which of these are ways of ameliorating either control or structural hazards in a scalar pipeline?:

a. A multi-ported memory  
b. Data forwarding  
c. A bubble/stall  
d. Delay slot

*Which of the following data hazards can be completely resolved in the MIPS pipeline described in the book using data forwarding (no bubble required)?

a. An R-type write to $t0 followed by an R-type read from $t0  
b. A load into $t0 followed by an R-type read from $t0  
c. A false dependency, or anti-dependency  
d. An R-type write to $t0 followed by an I-type read from $t0

*Which of the following hazards typically are of concern only for superscalar pipelines?:

a. Control hazards  
b. Structural hazards when the I-cache and D-cache are combined  
c. WAR hazards  
d. Structural hazards in the number of reservation stations
5. (10) Match the ten structures/techniques on the left with the goal of each on the right that fits it best (It should be a one-to-one mapping, just draw lines).

<table>
<thead>
<tr>
<th>Structure/Technique</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architectural register file (2)</td>
<td>1. Avoiding bubbles due to control hazards</td>
</tr>
<tr>
<td>Future file (8)</td>
<td>2. Enabling precise interrupts</td>
</tr>
<tr>
<td>History buffer (3)</td>
<td>3. Recovering from branch mis-predictions</td>
</tr>
<tr>
<td>Branch target buffer (9)</td>
<td>4. Avoiding extra pipeline stages for lw/sw</td>
</tr>
<tr>
<td>Predicated instructions (10)</td>
<td>5. Enabling speculation for static multi-issue</td>
</tr>
<tr>
<td>NaT or poison bit (5)</td>
<td>6. Simplifying pipeline control logic</td>
</tr>
<tr>
<td>Forced memory alignment (4)</td>
<td>7. Avoiding bubbles due to data hazards</td>
</tr>
<tr>
<td>Microcode (6)</td>
<td>8. Speculative register renaming</td>
</tr>
<tr>
<td>Branch prediction (1)</td>
<td>9. Avoiding unnecessary PC calculations</td>
</tr>
<tr>
<td>Data forwarding (7)</td>
<td>10. Getting rid of branches in the machine code</td>
</tr>
</tbody>
</table>

6. (5) Write "compiler", "assembler", or "CPU" next to each of the following indicating where it would typically be implemented (assume the compiler produces assembly code that must be assembled with the assembler).

- Static optimizations such as loop unrolling, constant folding, etc. **Compiler**
- Branch prediction **CPU**
- Resolving a symbol to a memory address **Assembler**
- Dynamic register renaming **CPU**
- Microcode **CPU**
7. (10) How many basic blocks are there in this MIPS assembly code snippet:

```mips
    gcd_wrapped:
        addi    $sp -4          # Reserve space
        sw      $ra 0($sp)      # Store $ra
        bne     $a1 $zero Recur
-----------------------------------------
        addi    $v0 $a0 0
        jr      $ra
-----------------------------------------

Recur:
        divu    $a0 $a1
        mfhi    $t0
        addi    $a0 $a1 0
        addi    $a1 $t0 0
        jal     gcd_wrapped
-----------------------------------------
        lw      $ra 0($sp)
        addi    $sp 4
        jr      $ra
```

*There are four basic blocks*

8. (5) What is the reorder buffer’s primary function in implementing precise interrupts?

*To commit instructions in order*

9. (5) What is a correlating branch predictor?

*Combines global and local branching behavior to make predictions*

10. (5) Traditional VLIW differs from EPIC as implemented on the Itanium in the way that instruction grouping is implemented, the former using static instruction windows and the latter using stops. Name an advantage of stops over static instruction windows.

*The stops are more forward compatible with future versions of the CPU that can issue more instructions at once.*

11. (5) The Pentium 4 allows up to 126 micro-operations to be outstanding at one time. Name one of the places a micro-operation might be stored while it is outstanding (issued but not yet committed) in a superscalar CPU (any superscalar CPU).

12. (5) The book identifies a particular property of caches for superscalar processors that is absolutely critical for IPC>1.0. What is this called?

*Non-blocking cache*