Part I. Regular Questions (hand in as hardcopy) [10 points total]

1. Vocabulary: (terms you need to know to discuss the subject intelligently) – Define the following terms using 1-3 sentences (and a diagram, if needed):
   a. Single Cycle Datapath [2 points each]
   b. Buffer Register
   c. Forwarding (in a pipeline)
   d. Multicycle Datapath
   e. MIPS Pseudoinstruction

Part II. MIPS Coding Problem (hand in as hardcopy) [30 points total]

2. Coding MIPS - Pointers: Assume that two integer variables \( x \) and \( y \) are pointed to by pointers (in the C language) respectively called \( p \) and \( q \), such that \( q - p = 24 \). Please translate the following C-code fragment into MIPS:

   ```c
   int *p, *q, *r, x = 10, y = 20, z;
p = &x; q = &y;
r = &x + 12;
*r = 32 - x;
q = *r + *p - y;
   ```
Document (comment) your code fully to get full credit and include a diagram of how and where each variable occurs in memory and what its contents are. Show how each pointer and variable value changes (or does not change) at each program step.

Part III. Datapath Tracing Problems (hand in as hardcopy)

3. Single Cycle Datapath Tracing: Given the following components of the single-cycle datapath, circle the components that are used for each instruction listed below, in the correct order, and whether the component is read from, written to, or both. Example: the jump instruction uses all the components in the Fetch partition, all the components in the Decode part except the small ALU, and so forth. (Here, Mux means multiplexer).

Fetch Partition: PC, small ALU for PC+4, Mux for selecting PCwrite, Instruction Memory

Decode Partition: Instruction Splitter, Sign Extender, Shifter, Register File, Small ALU (BTA)

Execute Partition: Mux to select ALU input B, Big ALU, Shifter, ALU output selector

Memory Partition: Data Memory, Write Enable line, Memory output Mux,

Please circle each of the above components that are used for each of the following independent instructions: \[10 \text{ pts} = 5 \text{ pts each}\]

add $s1, $s2, $s3

-and-

sw $s1, 227($s0)

Here is a picture to help you visualize the problem:
4. **Multi-cycle Datapath Tracing:** Given the following components of the multi-cycle datapath, circle the components that are used for each instruction listed below, in the correct order, and whether the component is read from, written to, or both. Example: the *jump* instruction uses all the components in the Fetch partition, all the components in the Decode part except the small ALU, and so forth. (Here, *Mux* means *multiplexer*).

**Fetch Partition:** PC, Mux for Memory, Memory, Memory Data Register (MDR)

**Decode Partition:** Instruction Register, Instruction Splitter, Mux for Register Address, Mux for Register Data

**Execute Partition:** Sign Extender, Shifter, Register File, “A” Buffer, “B” Buffer, “A” input Mux, “B” Input Mux, ALU, Shifter#2 (above ALU), ALUout buffer, ALU output Mux (to direct ALU output to Memory, RegisterFile, or PC)

Please circle each of the above components that are used for each of the following independent instructions: [10 pts = 5 pts each]

beq $s1, $s2, Load -and- addi $s1, $s2, 12345

Here is a picture to help you visualize the problem:

![Datapath Diagram]

**Part IV. Extra Credit** (hand in as *hardcopy*) [15 points total]

5. Modify the multi-cycle datapath (above) to support the jump-and-link instruction, for example, *jal SumVector*. Explain what hardware is added and how it works.