Before we begin, here is the diagram of the pipelined datapath you should reference:

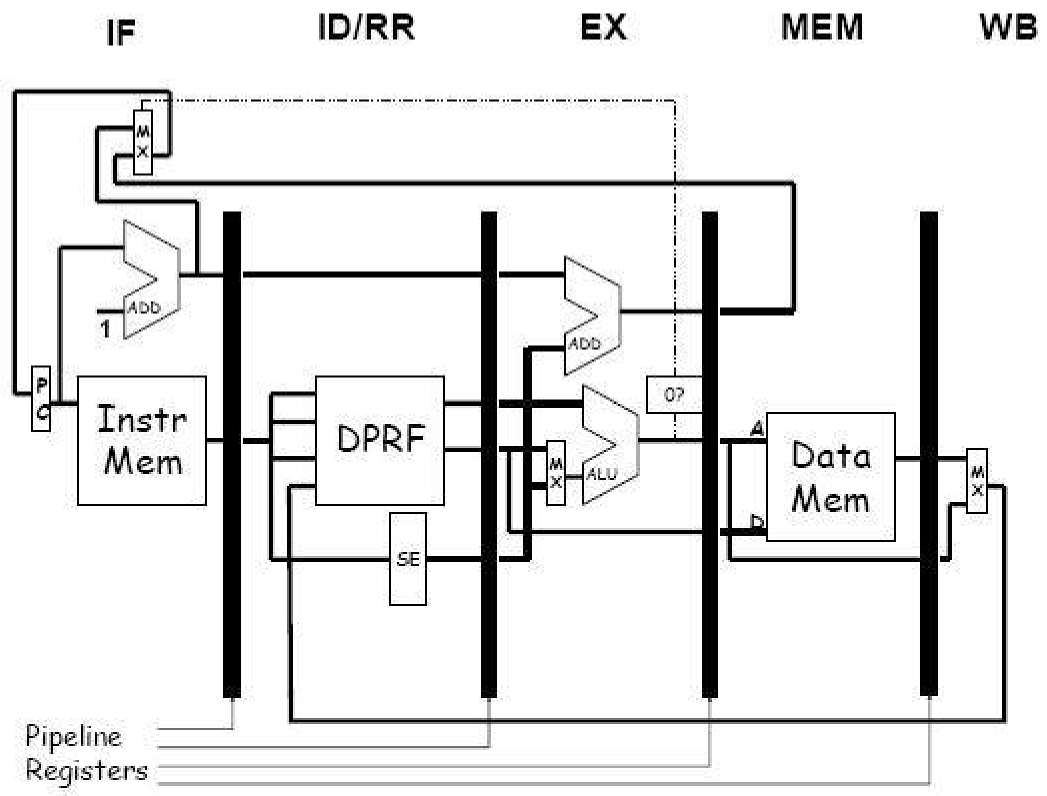

Figure 1: The Pipelined Datapath for LC-2200

Note that this is the same pipeline discussed in Chapter 5 Section 13.3 in your textbook.

Problem 1: Pipelining - Branch Prediction

1. [**10 points**] Regardless of whether we use the conservative approach or branch prediction (where we assume that the branch is not taken), explain why there is always a two cycle delay in the pipeline if the branch is taken (ie. why two NOPs are injected into the pipeline) before normal execution resumes.
2. [**10 points**] With reference to figure 1, identify and explain the role of the datapath elements that deal with the BEQ instruction. Give details as to what excatly happens cycle by cycle with respect to this datapath during the execution of a BEQ instruction. Assume that the processor is using the conservative approach to handle the control hazard. Your answer should include what happens when the branch is taken and when the branch is not taken.

Problem 2: Pipelining - Data Hazards

Note that you may ONLY assume the following:

- A LW instruction followed immediately by use of the loaded value incurs a one cycle stall.
- ADD/NAND/ADDI results are not available until their WB stage
- Branching uses the conservative branch prediction.

- Branch instructions can be completely determined (ie. if the branch is taken or not, as well as the branch's target address) in the EX stage.
- The processor for Problem 2 is NOT the same as the one in the diagram for Problem 1

Now, consider the following code fragment that increments the elements of an array stored in memory. At the beginning of each iteration, the register $a0 contains the address of the element to be incremented and $a1 contains the length of the array in bytes plus $a0. Here is the code listing:

```
LOOP: LW $t0, 0($a0)
      ADD $t0, $a0, $t0
      SW $t0,0($a0)
      ADDI $a0, $a0, 1
      BEQ $a1, $a0, DONE
      BEQ $zero, $zero, LOOP
DONE: HALT
```

1. [**12 points**]
   (a) Simulate the state of the pipieline for fifteen clock cycles and show which instruction is in each stage at each clock cycle. Use one line per cycle and one column per stage. Note that we do not know what was in the pipeline before the first ADDI instruction but assume that it does not cause any stalls. Furthermore, assume that the first branch is not taken for several iterations. Look in `answers.txt` for the appropriate format to use.
   (b) What is the average CPI (Cycles Per Instruction) for the loop?
   (c) If the processor operates at 1 GHz (that's 1×10^9 Hz), how long will it take to operate on an array of 5 million words?
2. [**12 points**]
   (a) Re-simulate the pipeline for 15 clock cycles where ADD/NAND/ADDI have full data-forwarding.
   (b) What is the average CPI for the loop?
   (c) If the processor operates at 1 GHz, how long will it take to operate on an array of 5 million words?
3. [**6 points**] Identify and count the number of:
   (a) Data Hazards
   (b) Strucural Hazards
   (c) Control Hazards

Problem 3: Process Scheduling

1. [**16 points**] Consider the following set of processes, with the length of the CPU burst time in milliseconds. The processes are assumed to have arrived in the order P1, P2, P3, P4, and P4 at time = 0.

Table 1: Processes, their Burst Times, and their Priorities

Process	Burst Time (ms)	Priority
P1	5	2
P2	4	5
P3	1	3
P4	6	1
P5	2	4

Draw four Gantt Charts illustrating the execution of these processes using FCFS (First Come First Serve), SJF (Shortest Job First), Non-Preemptive Priority (lower numbers = higher priority), and RR (Round Robin) with a time quantum of 1 (ignoring priority). Look at `answers.txt` for the appropriate format to use. Note that a process is no longer waiting once it has completed its burst time.

2. **[14 points]**

   (a) What is the waiting time for each process for each of the scheduling algorithms in part 1?

   (b) What is the turnaround time of each process for each of the scheduling algorithms in part 1?

   (c) Which of the scheduling policies in question 1 results in the minimum average waiting time over all processes?

   (d) Which scheduling algorithm has provably optimal average waiting time?

   (e) Which scheduling algorithm has the highest variance in turnaround time in general?

   (f) List the scheduling algorithm(s) which suffers from starvation.

   (g) Which of the above scheduling algorithms require a timer interrupt and preemption for their correct operation?

Problem 4: Memory Management

1. **[20 points]** Given memory partitions of 300 KB, 460 KB, 110 KB, 360 KB, and 700 KB (in that order), how would the first-fit, best-fit, and worst-fit algorithm place processes of size 200 KB, 350 KB, 420 KB, 100 KB, and 675 KB (in that order)? Which algorithm makes the most efficient use of memory? Note that if there isn't room for a process, state that it will not fit and it must wait. Multiple processes may end up in the same partition. Look in `answers.txt` for the appropriate format to use.