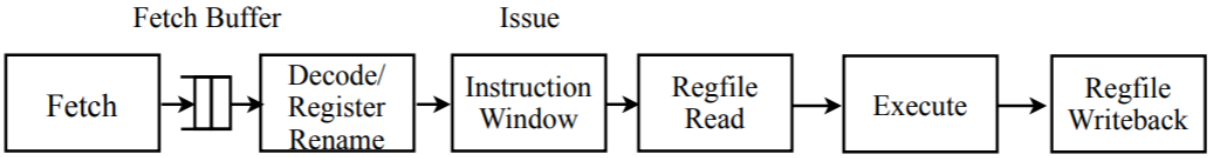
CS152 Section 10

Q1: Out-of-Order Pipeline Stages (2011 Quiz 3 Q1.B)

Consider a processor that uses a split issue window/ROB design, with a unified physical register file (similar to the MIPS R10k). A diagram of the processor is shown in the figure below.



Conceptually, an out-of-order pipeline can be organized into the following stages: Fetch, Decode, Register Rename, Dispatch, Issue, Execution, Completion, and Commit.

A. Which stage(s) allocate entries in the ROB?

B. At which stage(s) is an entry in the ROB deallocated?

C. At which stage(s) is an instruction allocated an entry in the issue window?

D. At which stage(s) is an instruction entry in the issue window deallocated?

E. At which stage(s) is a store entry allocated in the LD/ST queue?

F. At which stage(s) is a load entry allocated in the LD/ST queue?

G. At which stage(s) is a store performed (i.e., sent to the memory)?

H. At which stage(s) is a load performed (i.e., when is data returned that can be used by other dependent instructions)?

Q2: VLIW (2018 Midterm 2 Q4)

In this problem, we consider the execution of a code segment on a VLIW processor. The code we consider is the IMAX kernel, which finds the maximum value and its index in the list.

```
for (i = 0 ; i < N ; i++) {
      if (max < l[i]) {
            idx = i;
            max = l[i];
      }
}
```

```
# t0: i, s0: idx, f0: max, a0: N, a1: pointer of l[i]
loop:   fld f1, 0(a1)             # load l[i]
        flt.d t1, f0, f1          # set if max < l[i]
        fmax.d f0, f0, f1         # max = max < l[i] ? l[i] : max
        beqz t1, skip             # if max >= l[i], jump to skip
        addi s0, t0, 0            # update idx
skip:   addi a1, a1, 8            # bump l
        addi t0, t0, 1            # increment i
        bltu t0, a0, loop         # loop
```

Now we have a VLIW machine with five execution units:

- two ALU units, latency one cycle, also used for branch operations.
- one memory unit, latency two cycles, fully pipelined, each unit can perform either a store or a load.
- two FPU units, latency three cycles, fully pipelined, both can perform `flt.d` and `fmax.d`.

Assume there are no exceptions during the execution.

Schedule instructions for the VLIW machine with software pipelining but without loop unrolling in table below. Include the prologue and the epilogue. You do not need to find the optimal scheduling.

Label	ALU1	ALU2	MEM	FPU1	FPU2

Q3: Multithreading (2011 Quiz 4 Q3)

For this problem, we are interested in evaluating the effectiveness of multithreading using the following numerical code.

```
for (i = 0; i < N; i++) { // N = 1024
     S[i] = A[i] * B[i] + Y[i];
}
```

```
     addi $n, $0, 1024
     addi $i, $0, 0
loop:ld $a, A($i)
     ld $b, B($i)
     fmul $t, $a, $b
     ld $y, Y($i)
     fadd $s, $t, $y
     sd $s, S($i)
     addi $i, $i, 8
     addi $n, $n, -1
     bnez $n, loop
```

Assume the following:

- Our system does not have a cache.
- Each memory operation directly accesses main memory and takes 50 CPU cycles.
- The load/store unit is fully pipelined.
- After the processor issues a memory operation, it can continue executing instructions until it reaches an instruction that is dependent on an outstanding memory operation.
- The `fmul` and `fadd` instructions both have a use-delay of 5 cycles.

Q3.1: Round-Robin Scheduling

Suppose the pipeline is multithreaded. Threads are switched every cycle using a **fixed round-robin schedule**. If the thread is not ready to run on its turn, a bubble is inserted into the pipeline. Each thread executes the above code, and is calculating its own independent piece of the S array (i.e., there is no communication required between threads). Assuming an infinite number of registers, what is the **minimum** number of threads we need to fully utilize the processor? You are free to reschedule the assembly as necessary to minimize the number of threads required.

Q3.2: Data-Dependent Scheduling

Suppose threads are switched whenever there is a **data-dependent stall**. Assuming an infinite number of registers, what is the **minimum** number of threads we need to fully utilize the processor? You are free to reschedule the assembly as necessary to minimize the number of threads required.

Q4: Vectorization (2011 Quiz 4 Q2)

Indicate which of the following loops can be vectorized.

```
for (i = 0; i < N; i++)
     A[i] = A[i] + B[i];
```

```
for(i = 0; i < N; i++)
     A[i] = A[i+1] + B[i];
```

```
for(i = 0; i < N; i++)
     A[i] = A[i-1] + B[i]
```