



- ii) *Optimize the hardware.* The hardware team claims that it can improve the processor design to give it a clock rate of 600 MHz. Call this machine *Mopt*. The following measurements were made using a simulator for *Mopt*:

Instruction Class	CPI	Frequency
A	2	40
B	2	25
C	3	25
D	4	10

- a) What is the CPI for each machine?
  - b) What are the native MIPS rating for *Mbase* and *Mopt*?
  - c) How much faster is *Mopt* than *Mbase*?
3. (points 30) You are to improve cost/performance on an existing system based on a single-chip microprocessor having the following parameters.

Base Machine	Clock Frequency	100MHz	
	Die Size	10 mm x 10 mm	
	Instruction Mix	int	62
		FP	38
	CPI	Int	1.6
		FP	4.2

By using an optimizing compiler, the number of FP instructions is reduced by 20% and int instructions by 10% for the same application program.

- a) What is the new MIPS number of the base machine when using the optimizing compiler? (Instruction mix is changed!)
- b) When running the same application, how much performance gain do you expect from using such an optimizing compiler?

**By re-designing an FP hardware, die size will be increased by 20%, its FP CPI will be reduced to 2.8, clock frequency will remain as the same.**

- c) What is the new MIPS number of this improved machine with an ordinary compiler?

- d) Assuming die yield is proportional to the inverse of the cube of the die size, how much additional cost do you expect? Assume that the number of dies per wafer is inversely proportional to the die size and that all the cost is proportional to the die.

As an alternative, you are going use a new, scaled CMOS technology without re-designing.

In that case, the clock frequency changes to 125 MHz, its wafer cost doubles, and its die size is reduced by 15%. Use the same yield rule.

- e) How much performance improvement over the base machine do you expect?  
 f) How much additional cost over the base machine do you expect?

4. (points 20) The program below executes on the implementation also show below

```

add    r1, r2, r3
and    r4, r1, r5
sw     0(r4), r1
lw     r1, 8(r4)
xori  r5, r1, #1
beqz   r5, TARGET
sub   r5, r5, r5
.....

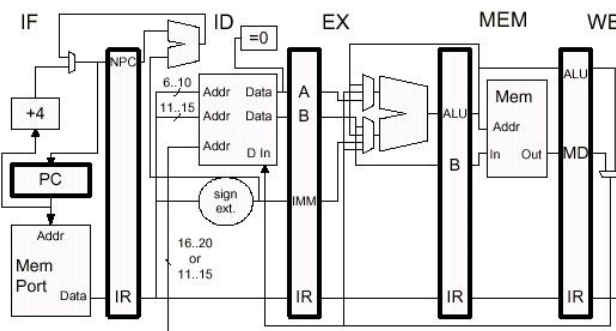
```

TARGET:

```

or     r10, r5, r1

```



The implementation includes only the forwarding paths that are shown in the figure. A new register value can be read in the same cycle it is written. Show a pipeline execution diagram for an execution of the code in which the branch is taken.