1. Students M, O & P are building custom hardware and compilers for their project

The following observations are known. CPI = 1

• Student O has built his design and it has a known execution time for a new program of 600 seconds.

• Student M proposes a single cycle data path, and plans to use a compiler that will generate 300 Billion dynamic instructions per execution.

• Student P proposes a pipelined data path, with a clock period of 1.2 ns. His compiler will generate 400 Billion instructions. Of these 400 Billion instructions, 25% are loads, and 20% are branches. 40% of all loads cause a 2-cycle load-use stall. The penalty for a mis-predicted branch is 5 cycles. The processor has full bypassing, and does not suffer from any other stalls.

How fast does M need to make his clock cycle in order to make the program run faster than O's?

1.

Assume that the M's cycle time is m.

There is:

$$300 \times 10^9 \times m < 600$$

$$m < 2 \times 10^{-9}\ s$$

i.e. M need to make his clock cycle less than 2 ns

2. Assume that you have a computer with 1 clock cycle per instruction (CPI=1) when all accesses to memory are in cache. The only accesses to data come from load and store instructions. Those accesses account for 25 % of the total number of instructions. Miss penalty is 50 clock cycles and miss rate is 5 %. Determine the speedup obtained when there is no cache miss compared to the case when there are cache misses

2.

CPU execution time = (CPU clock cycles + memory stall cycles) × cycle time = (# instructions × CPI + 0) × cycle time

= # instructions × cycle time.

Memory stall time = #instructions × (1 + 0.25) × 0.05 × 50

= #instructions × 3.125

CPU execution time no cache misse is:

(#instructions + #instructions × 3.125) × cycle time.

Therefore, the performance ratio is the rate of these execution time.

i.e. : $\frac{\text{CPU execution time no cache misses}}{\text{CPU execution time}} = 4.125$

Therefore, the processor with no cache misses is 4.125 time faster.

3. When Program X is run, the user CPU time is 3 seconds, the total wall clock time is 4 seconds, and the system performance is 10 MFLOP/sec. Assume that there are no other processes taking any significant amount of time, and the computer is either doing calculations in the CPU, or doing I/O, but it can't do both at the same time. We now replace the processor with one that runs six times faster, but doesn't affect the I/O speed. What will the user CPU time, the wall clock time, and the MFLOP/sec performance be now?

3. Because that CPUB performance/ CPUA performance
= CPUA time / CPUB time

$\Rightarrow \frac{6}{1} = \frac{3}{\text{CPU time}}$.
The user CPU time is 0.5 s.

Because the I/O time is not affected, it is 1s,
then the wall clock time is 1+0.5 = 1.5 s

Because the MFLOP
= number of floating point operations $\times 10^6$ / wall clock time
i.e. 10 = number of floating point operations $\times 10^6$/ 4
Number of floating point operations is $40/10^6$
And the new MFLOP/sec = $(40/10^6) \times 10^6 / 1.5$
= 26.667

4. For the 5-stage pipeline, with forwarding hardware discussed in class, our assumption was that the ALU is able to complete in one cycle because we assumed integer operations. Here, we assume that the ALU takes two cycles to complete. In other words, *the ALU is pipelined itself, making the entire pipeline 6 cycles*. The ALU only *generates a result after 2 cycles* (i.e., there is no way to extract any meaningful result after the ALU's first cycle). Memory (instruction and data) still returns data after one cycle. You may ignore branch and jump instructions in this problem.

Let's look at how this changes data hazards. For this question, we will examine only the ALU-ALU read after write (RAW) hazard where the two instructions are consecutive:

```
ADD x1, x2, x3        # x1 <= x2 + x3
ADD x5, x4, x1        # x5 <= x4 + x1
```

(i) Describe *how would you resolve this hazard with the minimum number of bubbles (if any) using a combination of data forwarding and stalls (if necessary)*.

(ii) Then fill the following timing diagram to illustrate how the pipeline will behave, and show where data forwarding happens, if at all, by drawing an arrow between the two stages that participate in it.

	CC0	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8

4.
(i). When the first ALU instruction completes the second ALU execute step, the second ALU instruction is already in the first ALU step. therefore only data forwarding is not suffice, it needs a NOP between the 2 instructions and a data forwarding betwee ALU 2 and the decode step.

(ii)

	CC0	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8
ADD x1, x2, x3	IF	ID	ALU1	ALU2	MEM	WB			
NOP									
ADD x5, x4, x1			IF	ID	ALU1	ALU2	MEM	WB	

5. Your favorite Computer is described with the following features:
- 95% of all memory accesses are found in the cache.
- Each cache block is two words, and the whole block is read on any miss.
- The processor sends references to its cache at the rate of 10^9 words per second.
- 25% of those references are writes.
- Assume that the memory system can support 10^9 words per second, reads or writes.
- The bus reads or writes a single word at a time (the memory system cannot read or write two words at once).
- Assume at any one time, 30% of the blocks in the cache have been modified.
- The cache uses write allocate on a write miss.

You are considering adding a peripheral to the system, and *you want to know how much of the memory system bandwidth is already used.*

(a) Calculate the percentage of memory system bandwidth used assuming the cache is Write Back.
(b) Calculate the percentage of memory system bandwidth used assuming the cache is Write Through.
Be sure to state your assumptions.

5. miss rate : 0.05 ; block size : 2 words.
frequency of memory operations from processor : 10^9
frequency of writes from processor = 0.25×10^9,
fraction of read hits : $0.75 \times 0.95 = 0.7125$
fraction of read misses : $0.75 \times 0.05 = 0.0375$
fraction of write hits : $0.25 \times 0.95 = 0.2375$
fraction of write misses : $0.25 \times 0.05 = 0.0125$

(a).
On a read hit there is no memory access.
On a read miss :
If replace line is modified, cache send two words to memory and memory send two words to the cache.
If replace line is clean, memory send two words to the cache.
On write hit there is no memory access, on a write miss :
If replaced line is modified, cache send two words to memory and memory must send two words to the cache.
If replace line is clean, memory send two words to the cache.

Therefore, average words transferred
$= 0.7125 \times 0 + 0.0375 \times (0.7 \times 2 + 0.3 \times 4) + 0.2375 \times 0$
$+ 0.0125 \times (0.7 \times 2 + 0.3 \times 4)$
$= 0.13$
Average bandwidth used = 0.13×10^9, the percentage of bandwidth used is 13%

(b)
On a read hit there is no memory access.
On a read miss, memory send two words to the cache. On a write hit the cache send a word to memory, on a write miss memory send two words to the cache, and the cache send a word to memory.

Therefore, average words transferred
$= 0.7125 \times 0 + 0.0375 \times 2 + 0.2375 \times 1 + 0.0125 \times 3$
$= 0.35$
The percentage of bandwidth used is 35%