1. Performance evaluation (18 %):

Assume that we have a workload with the instruction mix breakdown and CPIs for each type of instructions as shown in Table 1. The CPIs are measured assuming that the processor has the perfect cache (no cache misses).

Table 1: Instruction mix breakdown and CPIs

	Load	**Store**	**Integer**	**Floating Point**
Instruction mix breakdown	15%	10%	50%	25%
CPI under perfect cache	2	2	2	8

(a) (7%) Assume that the processor has the perfect cache. Given two kinds of processor improvements X and Y. X reduces the CPI of integer operations from 2 to 1, while Y reduces the CPI of floating point operations from 8 to 3. Which improvement achieves better performance?

(b) (7%) Assume the miss rate of instruction cache is 2%, and the miss rate of the data cache is 4%. The miss penalty is 60 cycles for all misses. A designer proposed two methods to improve cache performance. Method I reduces the miss latency to 20 cycles, and Method II improves the hit rate of both instruction and data cache to 1%. Which method achieves better performance?

(c) (4%) A designer improves system performance by parallelizing the execution of integer operations. Assuming no parallelization overhead, how much performance improvements can we get with 50 processors?

2. Cache (12%):

Assume that we have a 32KB 4-way set-associative cache with 32B block size.

(a) (3%) How many blocks and how many sets does the cache have?

(b) (3%) Given two memory addresses, 0x5FA84200 and 0x10248FFF, which sets are the two addresses mapped to?

(c) (6%) If we increase the associativity of the cache, which kind of cache misses can be reduced? If we apply the way-prediction technique on the cache, how could the technique improve cache performance?

3. Instruction and Pipeline (20%):

The following problems will use the C code shown below.

```
for(i=0; i != j; i++)
  b[i] = a[i] + 2;
```

Assume the variables i, and j are assigned to registers $s0 and $s1, and the base addresses of array a[] and b[] are stored in registers $s2 and $s3, respectively. Registers $s4, $s5, $s6 and $s7 are free to use.

(a) (6%) Translate the given C code to MIPS instructions. Your translation should be direct, without rearranging instructions to achieve better performance.

(b) (14%) Assume that the processor for executing the code has a traditional 5-stage 1-issue pipeline. The processor has perfect branch prediction and the branch is resolved in the decode stage.

i. (4%) Which kinds of pipeline hazard would happen when executing the code from problem (a)?

ii. (10%) If the loop exits after executing only two iterations, rearrange your code from problem (a) to achieve better performance. You may utilize the

loop unrolling technique.

4. (20%)

Three processes, P_1, P_2, and P_3 run their statements repeatedly in turn. That is, the system first executes P_1's S_1, then P_2's S_2, P_3's S_3, P_1's S_1, P_2's S_2, ...etc. Please use semaphores to enforce such running sequence between processes.

5. (5%+5%)

Please answer Yes /No and give brief explanations for following statements.

(a) The optimal page replacement algorithm is feasible.

(b) In UNIX, the name of a file is not stored in its inode.

6. (10%)

The memory access time is 12 ns. But if page fault occurs, the time **becomes** 4 ms. If we want the effective access time not over 20 ns, what is the maximum page fault rate?

7. (5%+ 5%)

The success of virtual memory comes from the ability to detect the needs of a process.

(a) How does OS know the needs?

(b) Only the ability to detect the needs is not enough for OS to run a program which size is larger than the physical memory.

What property of programs makes virtual memory feasible?