Uppsala Universitet
Instutitionen för informationsteknologi
Avdelningen för datorteknik

Kursens Namn: Datorarkitektur 2		Datum
Namn (efternamn först, v.g. texta)		Utbildningsprogram (eller liknande)
Namn (namnteckning)		Personnummer (10 siffror)
Termin och år då du först registrerade kursen	Inlämningstid	Bordsnummer

Lösta uppgifter (sätt kryss)		Erhållna poäng	Max poäng
1			4
2			4
3			6
4			4
5			8
6			8
7			8
8			8
9			8
10			8
Summa poäng:			66
Betyg:			-

Tentamen i Datorarkitektur 2

(1DT636, Datorarkitektur MN2 och 1IT190, Datorarkitektur IT)

Fredagen 2000-12-08

Place: Post Scriptum 1
Time: 9-14 (+10 minutes for course evaluation)
Allowed help: Calculator, pencil, and eraser.
Language: Swedish or English

General information

- The course evaluation will be collected and sealed in an envelope. It will not be looked at by me. Please fill it out after you are done with the exam. You will be allowed 10 bonus minutes to do so.
- If you find anything unclear, state your assumptions clearly
- I will turn up at 11:00 and 12:30 to answer questions.
- Solve at most one “problem” per page (there are 10 problems in total)
- Write your name on all pages
- Do not cross-reference between the solutions of different problems
- Not readable and/or not understandable answers result in zero points.
- Check the second page to see if you have earned any “bonus questions”. You will automatically be given maximum points for the corresponding bonus questions. If so, put a "**B**" in the column "lösta uppgifter" on the cover page.

May the force be with you,

// Erik Hagersten

I can be reached at 070-425 0502 during the exam.

Problem 1 (=Bonus 1), random questions (4p)

a) Show how the expression C := A + B is computed on a load/store architecture (2p)
b) Show how the expression C := A + B is computed on a stack-based architecture(2p)

Problem 2 (=Bonus 2), virtual memory system (4p)

The virtual memory system can be viewed as a "cache system". Explain what entity in the virtual memory system corresponds to:
a) the cache line size (2p)
b) the address tag comparison (2p)

Problem 3 (=Bonus 3), caches (6p)

a) What is a victim cache and what kind of cache misses can it potentially remove
(use the Mark Hill's miss categories all starting with the letter "C")? (2p)

b) What are *Spatial* and *Temporal* locality and why are they important for cache-based systems? (2p)

c) What kind/kinds of cache misses (Mark Hill's three "C" misses) are removed by adding:
   i) Twice as larger cache size?
   ii) LRU replacement instead of random replacement (One sentence each) (2p)

Problem 4 (=Bonus 4), hardware support (4p)

a) What special hardware is used to limit the latency of a virtual-to-physical translation lookup? Briefly describe how it works? (two sentence) (2p)
b) The virtual memory system may produces a precise exception. When does that happen and what is the difference between a precise and "imprecise" exception? (three sentences) (2p)

Problem 5, cache organization (8p)

a) Make a drawing of a physical cache with the following data
- Cache size: 2Mbyte
- Cache line size: 32byte
- Organization: 2-way
- CPU word size: 8 byte (this is the size of the data unit delivered from the cache to the CPU)
- Physical address size: 32 bits (i.e., 4Gbyte of address space can be addressed)

Clearly show the address bits ranges used for the indexing, the comparisons, and multiplexer (mux) selects, etc. Describe, either in words or in logic, the functionality of all logic needed to resolve a read access to the cache.(6p)

b) Which bit ranges would change if the cache size was increased to 8Mbyte while all the other parameters stayed the same? State the new bit ranges (you do not need to make completely new drawing) (2p)

Problem 6, loop scheduling (8p)

Consider the loop and the corresponding compiler-generated code

```
for (i=1; i<=1000; i++)
    x[i] = x[i] + s;

loop:   LD F0, 0(R1)        ;line 1
        ADDD F4, F0, F2     ;line 2
        SD 0(R1), F4        ;line 3
        SUBI R1, R1, #8     ;line 4
        BNEZ R1, loop       ;line 5
```

Assume that the array is stored in "backwards order" in the array, i.e., the address on x[i+1] is 8 smaller than the address of x[i].

The pipeline has the following characteristics	Delay
FP ALU op to another FP ALU op:	3 cycles
FP ALU op to store double:	2 cycles
LD double to FP ALU op:	1 cycles
INT ALU op to another INT ALU op	0 cycles
Branch delay slots:	1 cycles

a) Show where the bubbles are and write one-sentence comments for each line. How many cycles per iteration? (2p)
b) Show how the loop can be statically scheduled to improve the performance and calculate the number of cycles required for each iteration. How many cycles per iteration? (2p)
c) Show how loop unrolling with maximum optimizations could help avoiding <u>all</u> the stalls in this loop using the smallest amount of unrolling. You do not have to show the set-up/clean-up code before and after the loop. How many cycles per iteration? (2p)
d) Show how even more unrolling can help speed up the execution further on a superscalar DLX processor with one integer and one floating point unit. How many cycles per iteration? (2p)

Problem 7, optimizing for caches (8p)

Consider the following matrix multiplication code for X := Y * Z, where X, Y and Z are 1000x1000 matrixes of 64 bit elements.

```
for (i = 0; i < N; i = i + 1)
        for (j = 0; j < N; j = j + 1)
                {r = 0;
                 for (k = 0; k < N; k = k + 1)
                                r = r + y[i][k] * z[k][j];
                 x[i][j] = r;
                };
```

a) Rewrite the code to achieve a considerable improvement in cache utilization. (3p)
b) State the name of the used technique. (1p)
c) State a parameter setting that would suit a 1Mbyte cache well and explain why that is. (2p)
d) <u>Estimate</u> the expected speedup assuming
   - an infinitely large main memory with a 200 cycle latency,
   - a cache hit latency of 5 cycles,
   - and an iteration time of 20 cycles (if all accesses hit in the cache). (2p)

Problem 8, system performance (8p)

You have a system with the following properties

- 10% of the instructions are stores
- 20% of the instructions are loads
- 90% of the loads hit in the 1st level data cache
- 90% of the load accesses to the 2nd level cache hit there
- 80% of the stores hit in the 1st level data cache
- 75% of the store accesses to the 2nd level cache hit there
- All the instruction fetches hit in the separate L1 instruction cache
- The CPU is a "perfect" 1.0 CPI pipeline, if all the loads and stores hit in the 1st level cache.
- The CPU clock frequency is 100MHz, in-order, and has an infinitely large write buffer.
- There is a 10 cycle stall if you miss in L1 data cache and hit in the L2 cache.
- There is a 100 cycle stall if you miss in both L1 and L2 caches
- There is unlimited bandwidth at all levels of the hierarchy
- The cache line size is 64 bytes

a) What is the CPI for the system if the overhead in the memory system is also taken into account? (2p)

b) What is the bandwidth of all the memory traffic assuming a copy-back cache startegy and that 20% of the L2 misses causes a write-back (and that there is inclusion between the L1 and L2)? (2p)

c) What is the speedup if the second-level cache was made larger and thus its miss rate was cut in half?(2p)

d) What would be the speedup if instead the CPU was clocked twice as fast, but the memory and cache latency (measured in time) stayed the same? (2p)

Problem 9, I/O system (8p)

a) A disk is marked with an average seek time of 20ms, it rotates at 7200RPM and has a data transfer rate of 10Mbyte/s. What is the average time for accessing a file of 4kbyte? (4p)

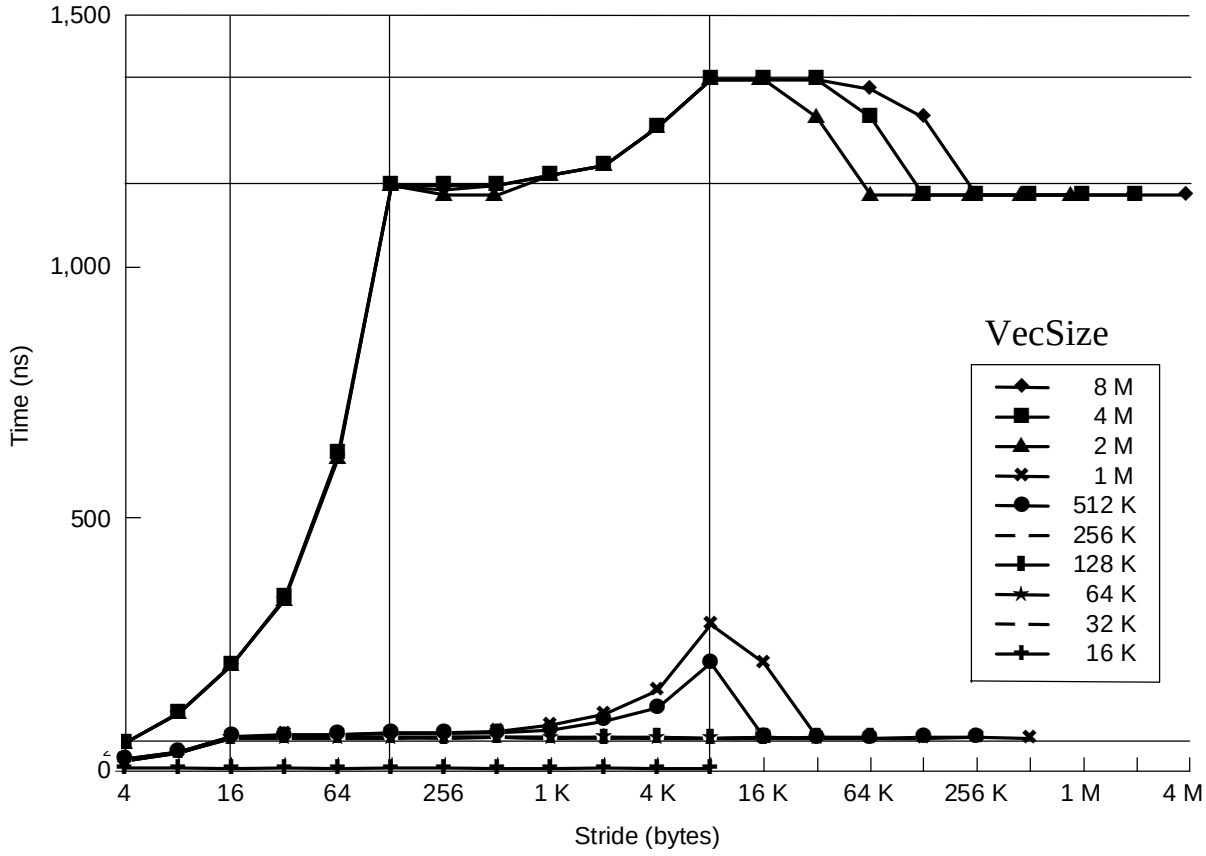
b) Describe the disk reads, disk writes and bit-logical operations needed for "small writes" in a RAID 5 system (2p)

c) What is the difference between an interrupt-driven and a DMA driven data transfer from an I/O device? (2p)

Problem 10, microbenchmarks (8p)

A computer architecture released 1993 has been characterized using a simple microbenchmark according to the graph below. The graph plots the average access time to an item in a vector of size *VecSize* stepped through with a stride *Stride*, as described in the code below:

```
for (times = 0; times < Max; time++)        /* do it many times*/
  for (i=0; i < VecSize; i = i + Stride)
    dummy = A[i];                           /* touch an item in the vector */
```



Answer each of these question with the requested number and a brief explanation of how you came to the conclusion:

a) What is the cache line sizes for the L1 and L2 caches? (2p)

b) What is the access time to the memory? (2p)

c) What is the page size? (2p)

d) How many entries are there in the TLB? (it is OK to answer with a range) (2p)