Umeå University
Department of Computing Science
5DV118 — Computer Organization and Architecture
Examination: April 13, 2012

Name (printed): _____

Swedish ID number: _____

Computer User-ID: _____

Signature: _____

Secret code number: _____

Instructions: / Instruktioner:

This examination will be graded anonymously. This page will be removed before the instructor receives the examination for grading. The secret code number given above must therefore be written on every answer page which you turn in to the examination proctor.

Denna skrivning rättas kodad. Detta blad kommer att avskiljas innan läraren får skrivningen för rättning. Ovanstående kod måste därför finnas på samtliga svarsblad när du lämnar skrivningen till skrivvakten.

To the proctor of the examination: / Till skrivningsbevakaren:

Detach this cover sheet from the examination and put it in the envelope which is addressed to Yvonne Löwstedt, Department of Computing Science.

Avskilj detta försättsblad och stoppa i kuvert som skickas till Yvonne Löwstedt, Datavetenskap.

Umeå University
Department of Computing Science
5DV118 — Computer Organization and Architecture
Examination: April 13, 2012

Secret code number: _____

1. Answers may be written in English or Swedish. However, all technical terms which do not have an absolutely standard representation in Swedish must be given in English.
2. A pocket engineering calculator (miniräknare) and English/X - X/English dictionary may be used. No other help materials are allowed.
3. Answers must be written on the official university answer sheets which are provided the sheets in numerical order of the problems, and write on only one side of the paper. Write only the question number and your secret code number on these pages; do not write your name or ID. Fields on the answer sheets: Kod: your secret code; Uppgift nr: problem number; Sidnr: page number; poäng: points (leave blank).
4. Show your work. For questions which require that an answer be computed, numerical answers without derivations will not receive credit. It is furthermore a good idea to show the symbolic formula used to obtain the answer, since that will result in more credit in the case that an error is made.
5. The examination has a total of 1000 points.
6. For problems with multiple parts, you have the choice, for each part, to do the problem, or to skip it for partial credit. In the table below, place an X in the position for any problem for which you have attempted a solution, and which you wish to have graded. It is extremely important that you fill in this table properly, because of the following option. For any box which is left blank, the associated question will not be graded, and you will instead be awarded 15% of the points for that question. Your decision to leave a box blank is definitive, so be very careful. For example, If you leave box 8(b) blank, your answer to that question will not be graded, even if it is completely correct. On the other hand, if you place an X in box 8(b), but provide no answer whatsoever to that question, you will not receive 15% of the points for that question. It is strongly recommended that you use a pencil, in case you change your mind!

Prob	1	2	3	4	5	6	7	8	9	10	11
(a)											
(b)					■						
(c)	■	■	■		■			■			■

(1: 100 points total) The average CPI (cycles per instruction) for different kinds of instructions for a computer is given in the table below, together with the frequency of each type of instruction for a given program.

Operation	Frequency	CPI
Fixed-point ALU	30%	2
Floating point	20%	10
Load	20%	4
Store	20%	4
Branch	10%	2

(a: 50 points) Compute how much faster (as a percentage) the program would run if the average CPI for floating-point operations were reduced to 5.

(b: 50 points) Suppose that the program is modified so that it contains only half as many floating-point instructions as the original program. The number of occurrences of each other type of instruction remains the same, as does its CPI value. Compute how much faster this new program is than the original one. (Hint: compute new frequency values for the instructions.)

(2: 100 points total) Consider a MIPS branch instruction of the following form, where `Label1` is a label which points to another instruction.

```
bne $t1, $t2, Label1
```

(a: 50 points) Illustrate how this instruction is represented, field by field, as a 32-bit word in MIPS. It is not necessary to give the opcode or numerical values for the register names, but all other information should be provided.

(b: 50 points) Explain precisely how the branch address is computed, using the information of part (a) as well as other critical values, such as the program counter. (Do not give details about how it is implemented in hardware; just give the formula for the final address in terms of register values, the program counter, etc.)

(3: 100 points total) Consider two functions in the language C which have the following prototypes:

```
int proc1(int i1, int i2);
int proc2(int i1, int i2, int i3, int i4, int i5, int i6);
```

Answer the following question in the specific context of the MIPS instruction set. Your answers must be specific to the MIPS architecture, and not generic, in order to obtain credit.

(a: 50 points) Assuming that the C type `int` is implemented as a single-precision, 32-bit integer, explain how the two parameters of `proc1` are passed from the calling program to `proc1`, as well as how the return value is passed back from `proc1` to the caller. Use the most efficient method possible, making use of registers whenever possible.

(b: 50 points) Repeat (a) for `proc2`, with emphasis upon the additional requirements imposed by the need to pass six distinct parameters from the caller to `proc2`.

Note (In case you do not know the syntax of C): The function `proc1` has two distinct call-by-value integer parameters, while `proc2` has six. Each function returns a single integer value.

(4: 60 points total) Answer the following questions about the representation of floating-point numbers using IEEE 754 format.

(a: 20 points) Identify the three fields which comprise this format and explain briefly the purpose of each.

(b: 20 points) Explain clearly, and illustrate by example, the notion of *overflow*.

(c: 20 points) Explain clearly, and illustrate by example, the notion of *underflow*.

(5: 40 points total) Explain why almost all digital computers use a complement representation (such as two's complement) for negative fixed-point numbers, rather than a sign-magnitude representation.

Note: This question is not about why two's complement is better than one's complement. It is about why both two's complement and one's complement are better design choices than sign-magnitude representation.

(6: 100 points total) An implementation of the MIPS architecture following the model developed in the textbook has the following delays for each of the five main stages:

Component	Delay
Instruction fetch (IF)	100 ps.
Instruction decode and register file read (ID)	70 ps.
Execution or address calculation (EX)	120 ps.
Data memory access (MEM)	100 ps.
Write back (WB)	40 ps.

(a: 30 points) With separate instruction and data memories, a non-pipelined implementation, and assuming all delays not explicitly identified above to be negligible, compute the latency for each of the five instructions `addi`, `bne`, `jr`, `lw`, and `sw`.

(b: 30 points) Repeat part (a), this time assuming a pipelined implementation with the five stages identified above.

(c: 40 points) Suppose that a less expensive implementation has a common memory for both instructions and data, with only one memory access per clock cycle possible. Explain how this would affect the answers given in parts (a) and (b). Be precise and quantitative in how the numbers would change.

(7: 100 points total) Answer the following questions relative to the five-stage pipeline architecture presented in the textbook and course slides.

(a: 35 points) Give an example sequence of two consecutive (in execution) MIPS instructions which illustrates a *read after write* (RAW) data dependency (called *read before write* in the slides), and which can be resolved completely by forwarding. Specify how many `nop` instructions must be inserted between the two instructions if no forwarding is employed, and also sketch briefly the forwarding logic which is necessary to eliminate these `nop`s.

(b: 35 points) Repeat (a), this time for a sequence which requires at least one `nop` instruction even with forwarding. Specify how many `nop` instructions must be inserted between the two instructions if no forwarding is employed and also when forwarding is employed, and also sketch briefly the forwarding logic which is necessary to eliminate unnecessary `nop`s.

(c: 30 points) Explain briefly the difference between a `nop` which is a *bubble* and one which is a *flush*. In particular, sketch how each is used to address hazard problems in pipelining.

(8: 100 points total) Consider a processor with an L1 cache for the MIPS instruction set and the following parameters.

Ideal CPI	6
Miss rate for access to instruction memory	1.5%
Miss penalty for access to instruction memory	120 clock cycles
Miss rate for access to data memory	2.5%
Miss penalty for access to data memory	135 clock cycles
Instruction mix	Load 10%, Store 15%, Arithmetic 65%, Branch 10%

(a: 50 points) Compute the actual CPI for this processor-cache pair.

(b: 50 points) Repeat (a), this time assuming that there is also a unified L2 cache with with a miss penalty of 20 clock cycles for access to this L2 cache from the L1 cache. Assume further that the miss rate for access to the L2 cache is 0.10%.

(9: 100 points total) A two-way set-associative cache is to hold 384K bytes of data. Assuming that a one-bit valid field is required (for each way), answer the following questions for one-word data blocks (per way), for words of 32 bits.

(a: 35 points) Determine the size of the tag field.

(b: 30 points) Compute the total number of bytes required for the cache.

(c: 35 points) Compute the size of the index needed for the cache.

(10: 100 points) A memory system operates on a bus which is one-word (32-bits) wide. It is governed by the following parameters:

Time to send the address to memory	1 clock cycle
Row cycle time	10 clock cycles
Column access time	4 clock cycles
Time to return one word from memory	1 clock cycle

(a: 40 points) Compute the number of clock cycles necessary to fetch one word from memory.

(b: 30 points) Compute the number of clock cycles necessary to fetch four words from memory, assuming that the memory is not interleaved, and that there are four blocks of one word each, with the words of each block in the same row, but the two blocks in different rows.

(c: 30 points) Compute the number of clock cycles necessary to fetch four words from memory, assuming that the memory is interleaved with two banks. Assume further that the four words are arranged into two blocks of two words each, with each word of a block in a different memory bank, but with the two blocks in different rows.

(11: 100 points total) A hard drive for a computer has the parameters given in the table below.

Parameter	**Value**
Rotational speed	5400 rpm
Average seek time	8 ms.
Controller overhead	1 ms.
Transfer rate	100M bytes/sec.

(a: 50 points) Compute the average amount of time required to read or write 100K bytes. Assume that the entire transfer requires only one seek and latency penalty.

(b: 50 points) Suppose that a given I/O task involves a large set of transfers of blocks of size 100K bytes. Each such transfer (of 100K bytes) requires 100000 instructions of the program running the task, as well as 50000 operating-system instructions. Assuming that all instructions take the same amount of time to execute and that the instructions are distributed uniformly over the entire time of each block transfer (incliding seek, latency, and overhead times), compute the minimum number of instructions per second which the processor must be able to execute in order to support this I/O operation, using a single disk drive of the specifications given above.

Note: It is to your advantage to show formulas with variables, not not just numbers, particularly for part (b). If you show formulas, you are far more likely to receive partial credit for part (b) if you make an error in computing the answer to part (a).