Exam in EDAF15 Algorithm Implementation

June 1, 2017, 14-19

The results will be announced through email on Saturday, June 3 at 15:00

Inga hjälpmedel!

You may answer in English, på svenska, auf Deutsch, или по-русски.

Examiner: Jonas Skeppstedt

30 out of 60p are needed to pass the exam.

1. (10p) **Pipeline**
   Speculative execution is important to achieve high performance in modern CPU's. What does it mean and how is it achieved? Which hardware is needed to make it work correctly and efficiently?

2. (10p) **Cache**

   - (2p) What do temporal and spatial locality mean?
   - (4p) What is meant by a 4-way associative cache?
   - (4p) Assume you have a direct mapped cache of total size 2^{15} bytes with a block size of 2^6 bytes, and a 32-bit address $A = (A_{31}, A_{30}, ..., A_3, A_2, A_1, A_0)$, where A_0 is the least significant bit. Which bits from A are used to select cache row, and which bits will then be stored as the tag for that cache block?

3. (20p) For the type `list_t`, implement the functions:

```
type struct list_t      list_t;
struct list_t {
        list_t*         next;
        void*           data;
};

list_t* new_list(void* data);
void free_list(list_t** list);
void insert_first(list_t** list, void* data);
void insert_last(list_t** list, void* data);
void reverse(list_t** list);
```

   - No special head node is used and an empty list is represented by `NULL`.
   - (3p) `new_list` should create a new list node with data pointer set to the parameter.
   - (4p) `free_list` should deallocate memory for the entire list but not data.
   - (4p) `insert_first` should insert a new list node first in the list.
   - (4p) `insert_last` should insert a new list node last in the list.
   - (5p) `reverse` should reverse the list, and update what the parameter points to to point to the new first node in the list. You are not allowed to allocate memory from the heap, with `alloca`, or with a variable-length array. You are only allowed to use local scalar local variables (such as pointers) allocated on the stack, i.e. normal local variables.
   - When you have modified the list in any way, you must make sure that what the `list` parameter points to points to the beginning of the list.

4. (6p) The following questions assume a two's complement representation of signed integers, and that there are 8 bits in a `signed char`.

   - (1p) How is -128 represented in a `signed char`?
   - (1p) What is the result if you try to divide -128 by 16 using arithmetic shift right 4?
   - (1p) How is -2 represented?
   - (1p) What is the result if you try to divide -2 by 16 using arithmetic shift right 4?
   - (2p) In general, how can arithmetic shift right be used to divide a signed integer by a power of two? Write a C function or explain in one sentence.

5. (4p) Explain what the following tools can be used for when analyzing the performance of a C program.

   - Cachegrind
   - Oprofile
   - Gcov
   - Gprof

6. (10p) Improve the performance of the `count_ones` function (see next page) which counts the number of ones in a matrix of `unsigned char`. The code should be valid C (i.e. no assembler or e.g. vector extensions are permitted). Motivate why you expect your optimizations are certain or likely to improve performance.

   Make the following assumptions:

   - The compiler makes no optimizations other than register allocation, and for instance all divide and multiply operations computed at runtime (taking 30 and 4 clock cycles, respectively). Other arithmetic integer instructions take one clock cycle. The latency of a load instruction is 2 clock cycles.
   - There are exactly 8 bits in an unsigned char (i.e. `CHAR_BIT` is 8).
   - The CPU is a 64-bit superscalar RISC processor (such as our POWER machine).
   - The function will be called a large number of times.
   - The numbers of rows and columns are typically very large (tens of thousands).
   - The distribution of ones and zeroes is not uniform. Expect at least ten times more zeroes than ones and that the ones are grouped like ”islands” if the matrix is interpreted as a map. You cannot make any particular assumptions about the sizes of such ”islands” though.
   - Code size is not important, except for the normal cost of instruction cache misses.
   - You may add help functions and data — and if you don’t want to type all such functions or data you are allowed to write a function which prints this for you.
   - The computer has enough RAM memory so the matrix fits in memory plus some more megabytes (i.e. RAM is not an issue unless you have a very memory consuming solution).
   - The first level caches (i.e. for data and instructions) are 64 KB each and the second level cache is 1 MB.

```c
#include <stdio.h>
#include <limits.h>

static size_t ones_in_char(unsigned char a)
{
        size_t  i;
        size_t  c;

        c = 0;

        for (i = 0; i < CHAR_BIT; i += 1) {

                if (a % 2 == 1)
                        c += 1;

                a /= 2;
        }

        return c;
}

size_t count_ones(size_t rows, size_t cols, unsigned char a[rows][cols])
{
        size_t  i;
        size_t  j;
        size_t  c;

        c = 0;

        for (j = 0; j < cols; j += 1)
                for (i = 0; i < rows; i += 1)
                        c += ones_in_char(a[i][j]);

        return c;
}
```

For example, with the following `main` function:

```c
int main()
{
        unsigned char   a[2][3] = { 1, 128, 3, 0xf };

        printf("%zu\n", count_ones(2, 3, a));

        return 0;
}
```

the output should be 8.

Lycka till och ha en riktigt skön sommar!