Exam in EDAF15 Algorithm Implementation

June 5, 2019, 14-19

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 and your grade is ⌊your score/10⌋.

There is always exactly one thread of execution so no thread level parallelization is useful.

1. (10p) **Pipeline**
   - (7p) To improve performance, superscalar processors perform speculative execution. What is that and which three essential parts of the processor are needed? What is the purpose of each? One point for each explained part.
   - (3p) What happens in the pipeline when the processor discovers that it is executing the wrong instructions due to mis-speculation?
2. (10p) **Cache**
   - (4p) Why are instruction and data caches not fully associative?
   - (3p) Consider what happens at a cache miss when some data is copied from the memory to the cache. A cache block size of one byte minimizes the risk of copying data that will not be used in the near future (i.e. useless data). It also minimizes the risk of overwriting data still in use. Why are cache blocks larger and what is a reasonable cache block size? (approximately)
   - (3p) Explain what a compromise between direct mapped and fully associative caches could be (not only could, it is very often used as well).
3. (10p) Write a function to reverse the nodes in a double linked non-circular list (i.e. the predecessor of the first node and the successor of the last node are null pointers). A list node is represented by:

```
typedef struct list_t list_t;

struct list_t {
        list_t*         succ;   /* pointer to the next node. */
        list_t*         pred;   /* pointer to the previous node. */
        void*           data;   /* pointer to data. */
};
```

   An empty list is represented by `NULL`.

```
void reverse(list_t** list)
{
        /* ... */
}
```

   After calling `reverse(&p)`, `p` should point to what previously was the last node. Trying to reverse an empty list should not crash the program.

4. (10p) Explain briefly the purpose of each of the following: the following:

   (a) (2p) function return address and in particular where it can be stored (hint: two different places).

   (b) (2p) register allocation

   (c) (2p) instruction scheduling

   (d) (2p) free-list

   (e) (2p) arena

5. (10p) Suppose we have four arrays `a`, `b`, `c`, and `d`, each with one million floating point numbers. Assume also that code usually needs the ith element from each array at approximately the same time such as in the frequently used code:

```
for (i = 0; i < N; i += 1)
        a[i] = b[i] + c[i] + d[i];
```

   What could motivate to declare these as:

```
#define N       (1000000)

double  a[N];
double  b[N];
double  c[N];
double  d[N];
```

   as opposed to:

```
#define N       (1000000)

struct type_t {
        double  a;
        double  b;
        double  c;
        double  d;
}       s[N];
```

   and then use:

```
for (i = 0; i < N; i += 1)
        s[i].a = s[i].b + s[i].c;
```

   Hint: the last 3p are quite difficult.

6. (4p) This question is similar to question 5, but instead uses parameters. Assume here that typical values of n more likely are at most a few hundred rather than a million. Assume the C file only consists of this function and it must be compiled to machine code separately (i.e. no link-time optimization as in lab 6).

```
void f(double* a, double* b, double* c, double* d, int n)
{
        int     i;

        for (i = 0; i < n; i += 1)
                a[i] = b[i] + c[i] + d[i];
}
```

and:

```
void f(type_t* s, int n)
{
        int     i;

        for (i = 0; i < n; i += 1)
                s[i].a = s[i].b + s[i].c + s[i].d;
}
```

Why would one or the other be preferable? Your motivation is more important than which one you prefer.

7. (6p) Write a function to compute a/b in C without using a machine instruction to divide, where $b = 2^k$, $k > 0$, and a and k are parameters. Write any implementation-defined behavior you rely on and assume it is true for the compiler you use. Motivate your code.

```
int f(int a, int k)
{

}
```