CS3101 Due: Friday 13th of February

Problem Set 1

CS3101 *Submission instructions on last page*

PROBBLEM 1. [40 points] The objective of this problem is to analyze cache misses for *merge-sort* algorithm:

`http://en.wikipedia.org/wiki/Merge_sort`

We rely on the ideal-cache model (see the lecture notes) and assume that that our ideal cache memory has Z words in total, with L words per cache-line. We assume that the input of our implementation of merge-sort is a one-dimensional array A consisting of n contiguous words. Since the pseudo-code in the wikipedia page of merge-sort uses lists, we shall consider the following document as our reference for the implementation:

`http://www.cs.mcgill.ca/ dprecup/courses/IntroCS/Lectures/comp250-lectur12.pdf`

Indeed, the algorithm there uses arrays. Here's another similar and good reference using arrays:

`http://highered.mcgraw-hill.com/sites/0070131511/student_view0/chapter2/algorithm_pseudoc`

We denote respectively by $T(n)$, $S(n)$ and $Q(n)$:

- the total number of arithmetic operations (including comparisons)
- the maximum of allocated space,
- the total number of cache misses

during the execution of our implementation of merge-sort, when applied to the array A of size n.

Question 1. [5 points] Recall $T(n)$ (no explanations required) and determine $S(n)$ (explanations required).

Question 2. [10 points] Determine a relation between Z, L and n for our implementation of merge-sort, when applied to A, to incur *cold misses only.* This relation does not need to be very precise.

In the rest of the problem, we assume that the relation determined in Question 2 is of the form

$$n < \alpha Z \tag{1}$$

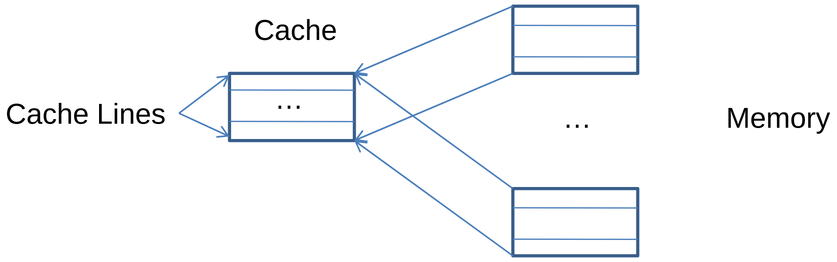
where α is a positive real number that does not depend on n or Z.

Question 3. [5 points] For $n \geq \alpha Z$ determine a recurrence relation satisfied by $Q(n)$.

Question 4. [15 points] Solve this recurrence relation in order to obtain an expression of $Q(n)$ that depends on Z, L, n and α.

Question 5. [5 points] Is merge-sort an algorithm which is suitable for implementation on modern desktops or laptops? Justify your answer.

PROBBLEM 2. [20 points] The following four questions are using this simple cache memory; the same as in class.



We recall its key features.

- Byte addressable memory.
- The Cache has size 32Kbyte with direct mapping and 64 byte lines (512 lines); so the cache can fit $2^9 \times 2^4 = 2^{13}$ `int`.
- **Therefore**, successive 32Kbyte memory blocks can line up in cache.
- A cache access costs **1 cycle while.** a memory access costs **100 cycles.**
- How addresses map into cache
  - Bottom 6 bits are used as offset in a cache line,
  - Next 9 bits determine the cache line

For each of the following four questions, answers must be justified. Total access times should be expressed in terms of S. That is, do **not** replace S by its numerical value.

Question 1. [5 points]

```
// sizeof(int) = 4 and Array laid out sequentially in memory
#define S ((1<<20)*sizeof(int))
int A[S];
// Thus size of A is 2^(20) x 16 bytes
```

```
for (i = 0; i < S; i++) {
     read A[2];
}
```

What is the total access time of this program? What kind of locality does it have, if any? What kind of cache misses?

Question 2. [5 points]

```
// sizeof(int) = 4 and Array laid out sequentially in memory
#define S ((1<<20)*sizeof(int))
int A[S];
// Thus size of A is 2^(20) x 16 bytes
for (i = 0; i < S; i++) {
     read A[i];
}
```

What is the total access time of this program? What kind of locality does it have, if any? What kind of cache misses?

Question 3. [5 points]

```
// sizeof(int) = 4 and Array laid out sequentially in memory
#define S ((1<<20)*sizeof(int))
int A[S];
// Thus size of A is 2^(20) x 16 bytes
for (i = 0; i < S; i++) {
     read A[(32 * i) % S];
}
```

What is the total access time of this program? What kind of locality does it have, if any? What kind of cache misses?

Question 4. [5 points]

```
#define S ((1<<19)*sizeof(int))
int A[S];
int B[S];
// Thus, in the main memory, the cache lines of
// B are just after all the cache lines of A
for (i = 0; i < S; i++) {
     read B[i], A[i];
}
```

What is the total access time of this program? What kind of locality does it have, if any? What kind of cache misses?

PROBBLEM 3. [40 points] In this problem, we consider the multiplication of two univariate polynomials with floating point number coefficients. Let $p(x)$ be such a polynomial of degree m

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_m x^m$$

with coefficients p_i, and let $q(x)$ denote a second such polynomial of degree n

$$q(x) = q_0 + q_1 x + q_2 x^2 + \cdots + q_n x^n$$

with coefficients q_i. Then we have

$$\begin{array}{rcl} p(x)q(x) & = & p_0q_0 + (p_0q_1 + p_1q_0)x + (p_0q_2 + p_1q_1 + p_2q_0)x^2 \\ & & +(p_0q_3 + p_1q_2 + p_2q_1 + p_3q_0)x^3 \\ & & +(p_0q_4 + p_1q_3 + p_2q_2 + p_3q_1 + p_4q_0)x^4 + \cdots \\ & & +(p_0q_{n+m} + p_1q_{n+m-1} + p_2q_{n+m-2} + \cdots + p_{n+m-1}q_1 + p_{n+m}q_0)x^{n+m}. \end{array}$$

Denoting $p(x)q(x)$ by

$$r(x) = p(x)q(x) = r_0 + r_1x + r_2x^2 + \cdots + r_{m+n}x^{m+n},$$

we see that the i-th coefficient of $r(x)$ can be expressed as

$$r_i = p_0q_i + p_1q_{i-1} + p_2q_{i-2} + \cdots + p_{i-1}q_1 + p_iq_0 = \sum_{j=0}^{i} p_j q_{i-j}.$$

In our pseudo-code below, we assume that each univariate polynomial of degree $d-1$

$$f(x) = q_0 + q_1x + q_2x^2 + \cdots + q_{d-1}x^{d-1}$$

is encoded by an array of length d,

$$[q_0, q_1, q_2, \ldots, q_{d-1}]$$

consisting of d contiguous words.

Question 1. [10 points] Let n and m be positive integers. The following pseudo-code multiplies two polynomials A and B (of C type *int) of degree $n-1$ and $m-1$ respectively, into a polynomial C of degree less than $n+m-2$.

```
void MultPoly1(A, n, B, m, C){
  for(i = 0; i < n+m-1; i++)
    C[i] = 0;
  for(i = 0; i < n; i++){
    for(j = 0; j < m; j++){
      C[i+j] = C[i+j] + A[i]* B[j];
    }
  }
}
```

1. We assume that n and m are large enough such none of A and B fits in cache. Give an estimate for $Q(n, m)$, the number of cache misses incurred by multPoly1, for an ideal (Z, L)-cache.

2. What are the practical consequences of this estimate?

Question 2. [30 points] The following pseudo-code can be used to multiply two polynomials A and B of degrees less than n, where n is a power of 2. To do this one should call multPoly2$(A, 0, n-1, B, 0, n-1, C, 0, 2n-2)$ after initializing C to the null vector. This algorithm divides each of A and B into "low and high bits", called `Alow`, `Ahigh`, `Blow`, `Bhigh`, and compute `Alow` $\times$ `Blow`, `Alow` $\times$ `Bhigh`, `Ahigh` $\times$ `Blow`, `Ahigh` $\times$ `Bhigh` recursively and construct C accordingly.

```
void multPoly2(A, i, j, B, k, l, C, p, q){
  int da = j - i;
  // int db = l - k;    // assume db = da holds
  // int dc = q - p;    // we should have dc = da + db

  if(da == 0)
    C[p] = C[p] + A[i] * B[k];
  else{
    m = (da + 1) / 2;    // observe that m >= 1 holds
    multPoly2(A, i, i+m-1, B, k, k+m-1, C, p, p+2m-2); // Alow * Blow
    multPoly2(A, i, i+m-1, B, k+m, l, C, p+m, p+3m-2); // Alow * Bhigh
    multPoly2(A, i+m, j, B, k+m, l, C, p+2m, q);       // Ahigh * Bhigh
    multPoly2(A, i+m, j, B, k, k+m-1, C, p+m, p+3m-2); // Ahigh * Blow
  }
}
```

Run this algorithm by hand with $A = [1, 0, 2, -1]$ and $B = [-1, 2, 0, 1]$ thus for $n = 4$. Check that the answer is correct. Then do each of the following tasks.

1. Give an estimate for $Q(n)$, the number of cache of misses incurred by `multPoly2` on input data of size n. To do so, observe that there exists a constant α such that

$(j-i+1)/Z < \alpha$ implies that multPoly2$(A, i, j, B, k, l, C, p, q)$ can be performed without any cache misses once the three array segments $A[i..j], B[k..l], C[p..q]$ have been loaded in cache.

2. Implement both multPoly1 and multPoly2 and demonstrate experimentally that multPoly2 runs faster. Ideally, you should make use of a performance tool analyzer such as VTune. However, on sufficiently large input data, it should clearly appear (simply by using the UNIX time command) that multPoly2 runs much faster than multPoly1.

Submission instructions.

Format: The answers to the problem questions should be typed.

- If these are programs, input test files and a `Makefile` (for compiling and running) are required.
- If these are algorithms or complexity analyzes, LaTeX is highly recommended; in any case a PDF file should gather all these answers.

All the files should be archived using the UNIX `tar` command.

Submission: The assignment should be returned to the instructor by email.

Collaboration. You are expected to do this assignment *on your own* without assistance from anyone else in the class. However, you can use literature and if you do so, briefly list your references in the assignment. Be careful! You might find on the web solutions to our problems that are not appropriate. For instance, because the parallelism model is different. So please, avoid those traps and work out the solutions by yourself. You should not hesitate to contact me if you have any questions regarding this assignment. I will be more than happy to help.

Marking. This assignment will be marked out of 100. A 10 % bonus will be given if your paper is clearly organized, the answers are precise and concise, the typography and the language are in good order. Messy assignments (unclear statements, lack of correctness in the reasoning, many typographical and language mistakes) may give rise to a 10 % malus.