2024年度入学試験

東京大学大学院情報理工学系研究科 コンピュータ科学専攻 試験問題

専門科目（問題1, 2）

2023年8月21日

13:00 – 14:10

注意事項

(1) 試験開始の合図があるまで, この問題冊子を開けないこと.

(2) 2題すべてに答えよ. 問題ごとに指定された解答用紙を使用すること.

(3) 解答用紙および問題冊子は持ち帰らないこと.

Specialized Subjects (Problems 1 and 2)

13:00 – 14:10, August 21, 2023

AY 2024 Entrance Examination

Department of Computer Science, Graduate School of Information Science and Technology

The University of Tokyo

Notice:

(1) Do not open this problem booklet until the start of the examination is announced.

(2) Answer the following 2 problems. Use the designated answer sheet for each problem.

(3) Do not take the problem booklet or any answer sheet out of the examination room.

下欄に受験番号を記入すること.

Write your examinee number in the box below.

受験番号	No.

問題 1

整数 $k > 0$ に対し，アルファベット $\Sigma = \{a, b\}$ 上の言語 L_k を以下のように定める．

$$L_k = \{x_1 \ldots x_n \in \Sigma^* \mid n \in \mathbb{Z} \wedge n \geq k \wedge x_{n-k+1} = a\}$$

ただし，$\mathbb{Z}$ は整数の集合，$x_i \in \Sigma$ とする．すなわち，L_k は後ろから k 番目の記号が a である語からなる言語である．

以下の問いに答えよ．

(1) L_3 を受理する非決定性有限オートマトンを与えよ．

(2) L_k を正則表現を用いて表せ．ただし，正則表現 r の i 回の連接を r^i と表記してよい．

(3) $L' = \bigcup_{k=1}^{\infty} L_{k^2}$ は正則言語か？そうであるなら L' を受理する有限オートマトンを与えよ．そうでないなら，L' が正則でないことを証明せよ．正則言語に対する反復補題を用いてよい．

(4) L_k を受理する全ての決定性有限オートマトンは，2^k 個以上の状態を持つことを証明せよ．

Problem 1

Given an integer $k > 0$, we define a language L_k over an alphabet $\Sigma = \{a, b\}$ by:

$$L_k = \{x_1 \dots x_n \in \Sigma^* \mid n \in \mathbb{Z} \wedge n \geq k \wedge x_{n-k+1} = a\}$$

Here, $\mathbb{Z}$ is the set of integers and $x_i \in \Sigma$. That is, L_k is the language that consists of words whose k-th symbol from the last is a.

Answer the following questions.

(1) Give a non-deterministic finite automaton that accepts L_3.

(2) Describe L_k using a regular expression. You may write the i-time concatenation of a regular expression r as r^i.

(3) Is $L' = \bigcup_{k=1}^{\infty} L_{k^2}$ a regular language? If so, give a finite automaton that accepts L'. If not, prove that L' is not regular. You may use the pumping lemma for regular languages.

(4) Prove that any deterministic finite automaton that accepts L_k has at least 2^k states.

問題 2

キャッシュラインサイズ（ブロックサイズ）が16バイトであり，全体で256バイトのデータを保持するダイレクトマップ方式のデータキャッシュを持つプロセッサPを考える．プロセッサPはデータキャッシュを介して，ロードワード命令lwによりメモリからデータを読み出し，ストアワード命令swによりメモリにデータを書き込む．ロードワード命令とストアワード命令のアドレスとデータのビット幅は32とする．データキャッシュのインデックスとオフセットは，メモリアドレスのビット表現を $a_{31}a_{30}\ldots a_0$ としたとき，$a_7a_6a_5a_4$ と $a_3a_2a_1a_0$ でそれぞれ与えられる．プロセッサPは，x0からx31までの32個の整数レジスタを持ち，x0は常に0を保持するゼロレジスタとする．

以下のプログラムSは，先頭アドレスが0x1000で長さ402の一次元配列Aに対して，大きさ3の平均化フィルタを適用し，その結果を先頭アドレスが0x2000で長さ400の一次元配列Bに格納する．配列AとBの各要素は32ビット符号付き整数とする．各命令の動作はプログラム中のコメント（#以降の記述）の通りとし，memory[addr]はaddr番地へのメモリアクセスを表す．レジスタx5, x6, x7の初期値をそれぞれ0x1640, 0x1000, 0x2000とする．

```
命令  0)        addi x2, x0, 3      # x2 <- x0 + 3
命令  1) Loop:  lw x8, 0 (x6)       # x8 <- memory[x6 + 0]
命令  2)        lw x9, 4 (x6)       # x9 <- memory[x6 + 4]
命令  3)        add x8, x8, x9      # x8 <- x8 + x9
命令  4)        lw x9, 8 (x6)       # x9 <- memory[x6 + 8]
命令  5)        add x8, x8, x9      # x8 <- x8 + x9
命令  6)        div x8, x8, x2      # x8 <- x8 / x2
命令  7)        sw x8, 0 (x7)       # memory[x7 + 0] <- x8
命令  8)        addi x6, x6, 4      # x6 <- x6 + 4
命令  9)        addi x7, x7, 4      # x7 <- x7 + 4
命令 10)        blt x6, x5, Loop    # if x6 < x5, goto Loop
```

以下の問いに答えよ．

(1) プロセッサP上でプログラムSを実行する場合のデータキャッシュのキャッシュヒット率を小数第3位まで求めよ．ただし，プログラムの実行開始時は，すべてのキャッシュラインが無効状態 (invalid) であるとし，プリフェッチャーはないものとする．

(2) プロセッサP上でプログラムSを実行する場合のIPC (instructions per cycle) を小数第3位まで求めよ．ただし，プロセッサPは，遅延なしでアクセス可能な命令メモリを持ち，命令フェッチおよび命令デコード，データのフォワーディングやレジスタファイルへのライトバックに遅延は発生しないものとする．また，毎クロックサイクル最大1命令，実行を開始し，各命令の完了まで後続命令を開始しない．命令add, addi, bltは1クロックサイクル，divは4クロックサイクルでそれぞれ実行される．命令lwとswは，キャッシュヒット時は1クロックサイクル，キャッシュミス時は4クロックサイクルで実行される．

(3) プログラムSを実行する場合のキャッシュヒット率を向上させるために，データキャッシュの構造にどのような変更を加えればよいかを，キャッシュヒット率が向上する理由と共に説明せよ．ただし，データキャッシュの容量は変えてはいけない．

(4) プロセッサPやデータキャッシュの構造を変更することなくキャッシュヒット率を向上させるための，プログラムSに対するソフトウェアレベルの最適化を具体例を用いて説明せよ．

Problem 2

Consider a processor P with a direct-mapped data cache that stores 256 bytes of data in total. The cache line size (block size) of the data cache is 16 bytes. Through the data cache, the processor P reads data from the memory by the load-word instruction `lw` and writes data to the memory by the store-word instruction `sw`. The address and data bit-widths of the load-word/store-word instructions are 32. When the bit representation of a memory address is $a_{31}a_{30}\ldots a_0$, the index and the offset of the data cache are $a_7a_6a_5a_4$ and $a_3a_2a_1a_0$, respectively. The processor P has 32 integer registers from `x0` to `x31`, and `x0` is the zero register that always keeps the value 0.

The following program S applies the average filter with size 3 to the one-dimensional array A with head address 0x1000 and size 402 and stores the result on the one-dimensional array B with head address 0x2000 and size 400. Each element of the arrays A and B is a 32-bit signed integer. The behavior of each instruction is described as a comment (the description after #) in the program, where `memory[addr]` represents a memory access to the address `addr`. The initial values of the registers `x5`, `x6`, and `x7` are 0x1640, 0x1000, and 0x2000, respectively.

```
Instruction  0)       addi x2, x0, 3     # x2 <- x0 + 3
Instruction  1) Loop: lw x8, 0 (x6)      # x8 <- memory[x6 + 0]
Instruction  2)       lw x9, 4 (x6)      # x9 <- memory[x6 + 4]
Instruction  3)       add x8, x8, x9     # x8 <- x8 + x9
Instruction  4)       lw x9, 8 (x6)      # x9 <- memory[x6 + 8]
Instruction  5)       add x8, x8, x9     # x8 <- x8 + x9
Instruction  6)       div x8, x8, x2     # x8 <- x8 / x2
Instruction  7)       sw x8, 0 (x7)      # memory[x7 + 0] <- x8
Instruction  8)       addi x6, x6, 4     # x6 <- x6 + 4
Instruction  9)       addi x7, x7, 4     # x7 <- x7 + 4
Instruction 10)       blt x6, x5, Loop   # if x6 < x5, goto Loop
```

Answer the following questions.

(1) Calculate the cache hit rate up to three places of decimals for the execution of the program S on the processor P. Suppose that every cache line of the data cache is invalid when the execution of the program starts, and there is no prefetcher.

(2) Calculate the IPC (instructions per cycle) up to three places of decimals for the execution of the program S on the processor P. Suppose that the processor P has an instruction memory with no access delay, and there is no delay on the instruction fetch, instruction decode, data forwarding, and write-back to the register file. The processor starts at most one instruction execution for every clock cycle. Until the instruction completes, the processor does not start the subsequent instructions. The processor executes each of `add`, `addi`, and `blt` instructions in one clock cycle, and executes `div` instruction in four clock cycles. The processor executes each of `lw` and `sw` instructions in one clock cycle in case of cache hits and four clock cycles in case of cache misses.

(3) Consider a modification to the data cache to improve the cache hit rate for the execution of the program S. Explain the modification with the reason why the cache hit rate is improved. Note that the data cache capacity should not be modified.

(4) Explain a software-level optimization of the program S, using a concrete example, for improving the cache hit rate without any modification to the processor P or the data cache.

2024年度入学試験

東京大学大学院情報理工学系研究科 コンピュータ科学専攻 試験問題

専門科目（問題3, 4）

2023年8月21日

14:55 – 16:05

注意事項

(1) 試験開始の合図があるまで, この問題冊子を開けないこと.

(2) 2題すべてに答えよ. 問題ごとに指定された解答用紙を使用すること.

(3) 解答用紙および問題冊子は持ち帰らないこと.

Specialized Subjects (Problems 3 and 4)

14:55 – 16:05, August 21, 2023

AY 2024 Entrance Examination

Department of Computer Science, Graduate School of Information Science and Technology

The University of Tokyo

Notice:

(1) Do not open this problem booklet until the start of the examination is announced.

(2) Answer the following 2 problems. Use the designated answer sheet for each problem.

(3) Do not take the problem booklet or any answer sheet out of the examination room.

下欄に受験番号を記入すること.

Write your examinee number in the box below.

受験番号	No.

問題 3

$G=(V,E)$（$|V|=n, |E|=m$）を自己ループ（1つの頂点だけを結ぶ辺）も多重辺（2つの頂点を結ぶ2本以上の辺）も含まない無向グラフとする．Gが連結グラフで，Gからある頂点vを抜くと，非連結になる場合，vをGのカット点と呼ぶ．

以下の問いに答えよ．

(1) Gが連結かどうかを判定するアルゴリズムを与え，時間計算量を評価せよ．

(2) 連結グラフGを入力とし，Gの全域木Tを出力する$O(m)$時間アルゴリズムを与えよ．

(3) Tを連結グラフGの全域木とし，vはTの葉以外の頂点（つまりvのT上の次数は2以上）であり，かつvはGのカット点ではないとする．また，eはvに接続するTの辺であるとする．eをGの別の辺$f \neq e$と置き換えることによってGの別の全域木がTから作れることを示せ．

(4) 連結グラフGを入力とし，Gのカット点をすべて求める$o(mn)$時間アルゴリズムを与えよ．

Problem 3

Let $G = (V, E)$ be an undirected graph with no self-loops (edges joining the same vertex) nor multi-edges (two or more edges joining the same two vertices), with $|V| = n, |E| = m$. If there is a vertex v in a connected graph G such that after deleting v, the resulting graph is not connected, we call v a cut vertex of G.

Answer the following questions.

(1) Describe an algorithm to check whether or not G is connected. Estimate the time complexity of the algorithm.

(2) Describe a $O(m)$ time algorithm to find a spanning tree T, given a connected graph G.

(3) Let T be a spanning tree of a connected graph G, and assume that v is a non-leaf node of T (i.e., the degree of v in T is at least two) and that v is not a cut vertex of G. Let e be an edge of T that is incident with v. Prove that one can obtain another spanning tree of G from T by replacing e with another edge $f \neq e$ of G.

(4) Describe a $o(mn)$ time algorithm, given a connected graph G, to find all cut vertices of G.

問題 4

C 言語風の値呼び戦略のプログラミング言語で記述された以下の関数 f を考える．ただし C 言語と異なり，整数データの範囲に制限はなく，オーバーフローは起きないものとする．

```
int f(int x)
{
  if (x<=0) return x+1;
  else return f(f(x-2));
}
```

例えば $f(1)$ の評価は以下のように行われ，返り値は 1，評価の際の関数 f の呼び出し回数は 3 である．

$$f(1) \longrightarrow f(f(-1)) \longrightarrow f(0) \longrightarrow 1.$$

以下の問いに答えよ．

(1) $f(2)$ を評価する際の関数 f の呼び出し回数を答えよ．

(2) すべての非負整数 n について，$f(n)$ の返り値が 1 であることを示せ．

(3) n を非負整数とする．$f(n)$ を評価する際の関数 f の呼び出し回数を n を用いて表せ．

(4) n を非負整数とする．値呼び戦略の代わりに名前呼び戦略を用いて $f(n)$ を評価する場合の関数 f の呼び出し回数を n を用いて表せ．

(5) 値呼び戦略だと停止しないが名前呼び戦略だと停止するようなプログラムの例を示せ．

Problem 4

Let us consider the following function f described in a C-like programming language with the call-by-value evaluation strategy. We assume that, unlike in the C language, there is no bound on integer data, and no overflow occurs.

```
int f(int x)
{
  if (x<=0) return x+1;
  else return f(f(x-2));
}
```

For example, $f(1)$ is evaluated as follows, where the return value is 1 and the number of calls of the function f during the evaluation is 3.

$$f(1) \longrightarrow f(f(-1)) \longrightarrow f(0) \longrightarrow 1.$$

Answer the following questions.

(1) Give the number of calls of the function f during the evaluation of $f(2)$.

(2) Show that the return value of $f(n)$ is 1 for every non-negative integer n.

(3) Let n be a non-negative integer. Express the number of calls of the function f during the evaluation of $f(n)$ in terms of n.

(4) Let n be a non-negative integer. Express the number of calls of the function f when $f(n)$ is evalulated by using the call-by-name strategy instead of the call-by-value strategy, in terms of n.

(5) Give an example of a program that does not terminate with the call-by-value strategy but terminates with the call-by-name strategy.