令 和 8 年 度

名古屋大学大学院情報学研究科
情報システム学専攻
入学試験問題（専門）

令和７年８月６日

注意事項

１．試験開始の合図があるまでは、この問題冊子を開いてはならない。

２．試験終了まで退出できない。

３．外国人留学生の志願者は、日本語と日本語以外の１言語間の辞書１冊に限り使用してよい。電子辞書の持ち込みは認めない。

４．日本語または英語で解答すること。

５．問題冊子、解答用紙６枚、草稿用紙３枚が配布されていることを確認すること。

６．問題は問１～８の８問がある。このうち６問を選択して解答すること。なお、選択した問題番号を解答用紙の指定欄に記入すること。

７．全ての解答用紙の所定の欄に受験番号を必ず記入すること。解答用紙に受験者の氏名を記入してはならない。

８．解答用紙に書ききれない場合は、裏面を使用してもよい。ただし、裏面を使用した場合は、その旨、解答用紙表面右下に明記すること。

９．解答用紙は試験終了後に６枚とも提出すること。

１０．問題冊子、草稿用紙は試験終了後に持ち帰ること。

問1

n を 2 以上の整数として，次の命題論理式 A_n, B_n, C_n を考える．

$$A_n = \bigwedge_{0 \leq i \leq n-2} (\neg x_i \vee x_{i+1})$$

$$B_n = \bigwedge_{0 \leq i \leq n-2} (y_i \supset y_{i+1})$$

$$C_n = \bigwedge_{0 \leq i \leq n-1} (y_i \supset x_i)$$

ここで，各 x_i と各 y_i は命題変数を表す．論理結合子の結合の強さは，強いものから順に否定 $(\neg)$，論理積 $(\wedge)$，論理和 $(\vee)$，含意 $(\supset)$ とする．各 P_i が命題論理式，m が 0 以上の整数であるとき，$\bigwedge_{0 \leq i \leq m} P_i$ は $P_0 \wedge \cdots \wedge P_m$ の略記である．以下の問いに答えよ．

(1) A_3 の真理値表を作成せよ．

(2) $A_3 \wedge x_0 \wedge \neg x_1$ は充足可能あるいは充足不可能のいずれであるか答えよ．

(3) A_n を真にする命題変数 x_i $(0 \leq i \leq n-1)$ への真理値割り当ての総数を n を使って表せ．

(4) $B_5 \wedge \neg y_0 \wedge y_4$ を真にする命題変数 y_i $(0 \leq i \leq 4)$ への真理値割り当ての総数を求めよ．

(5) $A_2 \wedge B_2 \wedge C_2$ を真にする命題変数 x_i と y_i $(0 \leq i \leq 1)$ への真理値割り当てをすべて求めよ．

Translation of technical terms

整数	integer	含意	implication
命題論理式	propositional formula	真理値表	truth table
命題変数	propositional variable	充足可能	satisfiable
論理結合子	logical connective	充足不可能	unsatisfiable
否定	negation	真	true
論理積	conjunction	真理値割り当て	truth assignment
論理和	disjunction		

問 2

(1)　$S = \{0, 1\}$ を値域とする確率変数 X_1, X_2 を考え，$Y_1 = (X_1 + X_2) \bmod 2$ として確率変数 Y_1 の値を定める．また，写像 $f : S \times S \to S$ に対し $Y_2 = f(X_1, X_2)$ として確率変数 Y_2 の値を定める．

(a) X_1, X_2, Y_1, Y_2 の結合エントロピー $H(X_1, X_2, Y_1, Y_2)$ が，X_1, X_2 の結合エントロピー $H(X_1, X_2)$ と等しくなることを示せ．

(b) X_1, X_2 と Y_1, Y_2 との相互情報量 $I(X_1, X_2; Y_1, Y_2)$ が，Y_1, Y_2 の結合エントロピー $H(Y_1, Y_2)$ と等しくなることを示せ．

(c) 相互情報量 $I(X_1, X_2; Y_1, Y_2)$ を最大化する写像 f の例を 1 つ示し，例示した f により得られる $I(X_1, X_2; Y_1, Y_2)$ が最大化されている理由を説明せよ．

(2)　以下の生成行列 G とパリティ検査行列 H を持つ $(7, 4)$ ハミング符号 C を用い，ビット誤り確率 p（ただし $p < 0.5$ とする）の 2 元対称通信路における誤り訂正を行う．

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & \boxed{\quad a \quad} & & \end{pmatrix}, H = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

(a) 生成行列 G の枠線 a の部分に記載すべき成分を示せ．

(b) C の符号語の 1 つを一様ランダムに選択して送信したところ，系列 1110101 が受信された．この系列に対して誤り訂正を行い，送信されたと推測される符号語を示せ．

(c) 符号語 0000000 を送信したとき，1 ビット誤り訂正復号の結果が正しく 0000000 となる確率を求めよ．

Translation of technical terms

値域	range	確率	probability
確率変数	random variable	2 元対称通信路	binary symmetric channel
写像	mapping	誤り訂正	error correction
結合エントロピー	joint entropy	成分	component
相互情報量	mutual information	符号語	codeword
最大化	maximize	一様ランダム	uniformly at random
生成行列	generator matrix	送信	transmit
パリティ検査行列	parity check matrix	系列	sequence
ハミング符号	Hamming code	受信	receive
ビット誤り	bit error	誤り訂正復号	error-correction decoding

問3

S を整数の空でない有限集合とし，$|S|$ は S に含まれる要素の個数を表すとする．また，S は1次元整数配列に格納されることで表現され，S を表現する配列の要素はインデックスに関して整列されているとは限らないとする．配列のインデックスは1から始まり，配列の各要素の参照は $O(1)$ で実行できると仮定する．

k を $1 \leq k \leq |S|$ を満たす整数とする．S の k 番目に小さい要素とは「S の要素のうち小さい方から数えて k 番目の要素」のことである．S の k 番目に小さい要素を求める方法について，以下のすべての問いに答えよ．なお，計算量を解答する問いでは，O 記法（オーダー）で最も小さい漸近的計算量を解答すること．例えば，n をパラメータとするとき，$O(n)$ は $O(n^2)$ よりも小さい．

(1) S の要素の最小値を求めるアルゴリズムのうち，最大時間計算量が最も小さくなるアルゴリズムの最大時間計算量を $|S|$ に関する O 記法を用いて答えよ．

(2) S の要素の最大値を求めるアルゴリズムのうち，最大時間計算量が最も小さくなるアルゴリズムの最大時間計算量を $|S|$ に関する O 記法を用いて答えよ．

(3) S の k 番目に小さい要素は S を表現する1次元整数配列をインデックスに関して昇順に並び替えて得られた配列 $\mathtt{s}$ の k 番目の要素 $\mathtt{s}[k]$ を取り出せば求められる．大小比較に基づく整列アルゴリズムを並び替えに用いた場合，この求め方の最大時間計算量を最も小さくするにはどのような整列アルゴリズムを用いればよいか，用いる整列アルゴリズムの名称およびその最大時間計算量を $|S|$ に関する O 記法を用いて答えよ．

(4) S の k 番目に小さい要素を求める方法の1つである次ページのアルゴリズム $\mathsf{select}(S, k)$ について，以下のすべての問いに答えよ．なお，空でない有限集合 X の中央値は X の $\lceil \frac{|X|}{2} \rceil$ 番目に小さい要素とする．ここで，$\lceil \cdot \rceil$ は天井関数である．

(a) $|S| = 10m + 5$（ただし，m は非負整数）としたときのアルゴリズム $\mathsf{select}(S, k)$ の手順5. の有限集合 A に含まれる要素数の最大値を以下の空欄（ア），（イ），（ウ）に正整数を埋めて $|S|$ の式で表せ．

$$|A| \text{ の最大値} = \frac{\boxed{\text{（ア）}}\,|S| - \boxed{\text{（ウ）}}}{\boxed{\text{（イ）}}}$$

(b) 整数の空でない有限集合 S と $1 \leq k \leq |S|$ を満たす整数 k に関して $\mathsf{select}(S, k)$ の実行の最大時間計算量を $|S|$ に関する O 記法を用いて答えよ．なお，手順2. の部分集合 $G_1, G_2, \ldots, G_{\lceil \frac{n}{5} \rceil}$ の計算，手順5. の集合 A，B の計算の最大時間計算量はそれぞれ $O(|S|)$ であると仮定してよい．また，$0 < c < 1$ を満たす定数 c と任意の正整数 i について以下の式が成り立つ．

$$n + c^1 n + c^2 n + \cdots + c^i n \leq \frac{1}{1-c} n$$

アルゴリズム $\mathsf{select}(S, k)$

入力 整数の有限集合 S（$|S| > 0$），非負整数 k（$1 \leq k \leq |S|$）

出力 S の k 番目に小さい要素

手順

1. $|S| \leq 5$ のときは S の k 番目に小さい要素を返して終了する．そうでないときは 2. へ進む．
2. 以下のすべてが成り立つように S の部分集合（ぶぶんしゅうごう） $G_1, G_2, \ldots, G_{\lceil \frac{|S|}{5} \rceil}$ を作る．
   - $S = G_1 \uplus G_2 \uplus \cdots \uplus G_{\lceil \frac{|S|}{5} \rceil}$
   - $|G_1| = |G_2| = \cdots = |G_{\lceil \frac{|S|}{5} \rceil - 1}| = 5$
   - $1 \leq |G_{\lceil \frac{|S|}{5} \rceil}| \leq 5$

   なお，$\uplus$ は非交和（ひこうわ），つまり，$X = Y \uplus Z$ は $X = Y \cup Z$ かつ $Y \cap Z = \emptyset$ を意味する．
3. M を $G_1, G_2, \ldots, G_{\lceil \frac{|S|}{5} \rceil}$ それぞれの中央値からなる集合，すなわち，$\{m_i \mid 1 \leq i \leq \lceil \frac{|S|}{5} \rceil,\ m_i \text{ は } G_i \text{ の中央値} \}$ とする．
4. M の中央値 x を $\mathsf{select}(M, \lceil \frac{|M|}{2} \rceil)$ により求める．すなわち，$x = \mathsf{select}(M, \lceil \frac{|M|}{2} \rceil)$ とする．
5. S の要素のうち x より小さいものからなる集合を A，x より大きいものからなる集合を B とする．すなわち，$A = \{a \in S \mid a < x\}$，$B = \{b \in S \mid b > x\}$ とする．
6. $|A| = k - 1$ のときは x を，$|A| \geq k$ のときは $\mathsf{select}(A, k)$ の結果を，$|A| < k - 1$ のときは $\mathsf{select}(B, k - |A| - 1)$ の結果を返して終了する．

Translation of technical terms

整数	integer	最小値	minimum
空	empty	アルゴリズム	algorithm
有限集合	finite set	最大時間計算量	worst-case time complexity
要素	element	最大値	maximum
1 次元整数配列	one-dimensional integer array	整列アルゴリズム	sorting algorithm
格納する	store	昇順	ascending order
インデックス	index	並び替える	sort
整列する	sort	大小比較	magnitude comparison
参照	reference	中央値	median
実行する	run	天井関数	ceiling function
計算量	complexity	非負整数	non-negative integer
O 記法	big O notation	正整数	positive integer
オーダー	order	定数	constant
漸近的計算量	asymptotic complexity	部分集合	subset
パラメータ	parameter	非交和	disjoint union

問4

有限オートマトン A は，5項組 $\langle Q, \Sigma, R, I, F\rangle$ で与えられる．ここで，Q は状態の有限集合，Σ はアルファベットの集合，I $(\subseteq Q)$ は初期状態の集合，F $(\subseteq Q)$ は最終状態の集合である．また，R $(\subseteq Q \times \Sigma \times Q)$ は状態遷移を表す関係である．

$w \in \Sigma^*$ と関係 R から定まる関係 $R[w]$ $(\subseteq Q \times Q)$，A が受理する言語 L_A，ならびに，言語 M $(\subseteq \Sigma^*)$ から定まる言語 $\mathrm{HF}_\Sigma(M)$ を次のように定める．

$$R[w] = \begin{cases} \{(q,q) \mid q \in Q\} & \cdots \quad w = \varepsilon \text{ (空列) のとき} \\ \{(q,q'') \mid (q,q') \in R[u],\ (q',a,q'') \in R\} & \cdots \quad w = ua \ (u \in \Sigma^*,\ a \in \Sigma) \text{ のとき} \end{cases}$$

$$L_A = \{w \in \Sigma^* \mid R[w] \cap (I \times F) \neq \emptyset\}$$

$$\mathrm{HF}_\Sigma(M) = \{w \mid \exists u \in \Sigma^*\ |u| = |w| \wedge wu \in M\}$$

ここで，$|u|$ は $u \in \Sigma^*$ の長さを表す．このとき，(1) から (4) に答えよ．

(1) 有限オートマトン $A_e = \langle Q_e, \{a,b\}, R_e, \{p_1\}, \{p_2\}\rangle$ について次の問いに答えよ．ここで，$Q_e = \{p_1, p_2, p_3\}$, $R_e = \{(p_1,a,p_2), (p_3,a,p_2), (p_2,b,p_3)\}$ とする．

(a) $R_e[a]$, $R_e[aa]$, $R_e[aba]$ をそれぞれ示せ．

(b) L_{A_e} を正規表現を用いて示せ．

(2) $\mathrm{HF}_{\{a,b\}}(\{\varepsilon,\ aaa,\ abbb,\ baaaab\})$ を示せ．

(3) $w \in \Sigma^*$ に対して，$w_\perp$ は $|w| = |w_\perp|$ を満たす文字列 $\perp \cdots \perp$ を表すとする．有限オートマトン $A_1 = \langle Q, \Sigma, R_1, \{q_1^I\}, \{q_1^F\}\rangle$ が与えられたとき，有限オートマトン $A_2 = \langle Q, \{\perp\}, R_2, \{q_2^I\}, \emptyset\rangle$ が，任意の状態 $q \in Q$ に対して次の (i) と (ii) を共に満たすように A_2 を定義せよ．具体的には，与えられた $A_1 = \langle Q, \Sigma, R_1, \{q_1^I\}, \{q_1^F\}\rangle$ を用いて R_2 および q_2^I を示せ．

(i) 任意の $w \in \Sigma^*$ について

$$(q, q_1^F) \in R_1[w] \text{ ならば } (q_2^I, q) \in R_2[w_\perp]$$

(ii) 任意の $v \in \{\perp\}^*$ について $v = w_\perp$ を満たす $w \in \Sigma^*$ が存在して

$$(q_2^I, q) \in R_2[v] \text{ ならば } (q, q_1^F) \in R_1[w]$$

(4) (3) の有限オートマトン A_1 と A_2 を用いて，$A_3 = \langle Q \times Q, \Sigma, R_3, \{(q_1^I, q_2^I)\}, F_3\rangle$ を定める．

$$R_3 = \{((q_1,q_2), a, (q_1',q_2')) \mid (q_1,a,q_1') \in R_1, (q_2,\perp,q_2') \in R_2\}$$
$$F_3 = \{(q,q) \mid q \in Q\}$$

このとき，任意の $w \in \Sigma^*$ について以下の性質が成立することを証明せよ．

$$w \in \mathrm{HF}_\Sigma(L_{A_1}) \text{ かつ，そのときに限り } w \in L_{A_3}$$

Translation of technical terms

有限	finite	関係	relation
オートマトン	automata	受理する	accept
5 項組	5-tuple	言語	language
状態	state	空列	empty string
集合	set	長さ	length
アルファベット	alphabet	正規表現	regular expression
初期状態	initial state	文字列	string
最終状態	final state	任意の	arbitrary
状態遷移	state transition	存在する	exist

出題意図と解答例

本問はオートマトンの基礎的概念の理解，形式的な定義の読解力，および，証明能力を問うことを出題意図としている．

(1) (a) $R_e[a] = \{(p_1, p_2), (p_3, p_2)\}$
$R_e[aa] = \emptyset$
$R_e[aba] = \{(p_1, p_2), (p_3, p_2)\}$

(b) $a(ba)^*$ など

(2) $\mathrm{HF}_{\{a,b\}}(\{\varepsilon,\ aaa,\ abbb,\ baaaab\}) = \{\varepsilon,\ ab,\ baa\}$

(3) $R_2 = \{(q', \bot, q) \mid (q, a, q') \in R_1\}$
$q_2^I = q_1^F$

(4) 任意の $w \in \Sigma^*$ について，$w \in \mathrm{HF}_\Sigma(L_{A_1}) \implies w \in L_{A_3}$ を証明する．

$w \in \mathrm{HF}_\Sigma(L_{A_1})$ とすると，HF の定義より $|v| = |w|$ かつ $wv \in L_{A_1}$ を満たす $v \in \Sigma^*$ が存在する．よって $(q_1^I, q_1^F) \in R_1[wv]$ であるから，$(q_1^I, q) \in R_1[w]$ かつ $(q, q_1^F) \in R_1[u]$ を満たす $q \in Q$ が存在する．このとき，(3)(i) の性質より $(q_2^I, q) \in R_2[u_\bot]$ が成立する．
ここで $|u| = |w|$ より $u_\bot = w_\bot$ であるから，$(q_2^I, q) \in R_2[w_\bot]$ がいえ，R_3 の構成から $((q_1^I, q_2^I), (q, q)) \in R_3[w]$ が成り立つ．よって $w \in L_{A_3}$ が成り立つ．

次に，任意の $w \in \Sigma^*$ について，$w \in L_{A_3} \implies w \in \mathrm{HF}_\Sigma(L_{A_1})$ を証明する．

$w \in \Sigma^*$ とすると，$((q_1^I, q_2^I), (q, q)) \in R_3[w]$ が成り立つ．よって R_3 の構成から $(q_1^I, q) \in R_1[w]$ かつ $(q_2^I, q) \in R_2[w_\bot]$ が成り立つ．このとき，(3)(ii) の性質より $w_\bot = u_\bot$ を満たす $u \in \Sigma^*$ が存在して $(q, q_1^F) \in R_1[u]$ が成立する．よって $(q_1^I, q_1^F) \in R_1[wu]$ が成り立つ．
ここで $|u| = |w|$ かつ $wu \in L_{A_1}$ であるから，$w \in \mathrm{HF}_\Sigma(L_{A_1})$ が成り立つ．

問 5

表 1 は、入力{I_0, I_1}、状態{Q_0, Q_1, Q_2, Q_3, Q_4, Q_5, Q_6}、出力{X_0, X_1} からなる完全定義順序回路の状態遷移表および出力表である。以下の(1)と(2)の問いに答えよ。

(1) 表 1 に対応する完全定義順序回路に対して 1-等価の同値類（1-等価の状態のグループ）を示せ。

(2) 上記 (1) で導出した 1-等価の同値類から順に状態集合を k-等価の同値類にグループ化し、状態数を最小化せよ。2-等価以降、状態数が最小になるまでの過程が分かるようにグループ化の途中経過を全て示せ。また、最小化後の状態遷移表と出力表を示し、最小化の過程と最小化前後の状態の対応関係を明示せよ。

表 1 完全定義順序回路の状態遷移表と出力表

	入力	
現状態	I_0	I_1
Q_0	Q_4, X_0	Q_5, X_0
Q_1	Q_6, X_0	Q_2, X_0
Q_2	Q_3, X_0	Q_0, X_1
Q_3	Q_1, X_0	Q_4, X_0
Q_4	Q_5, X_0	Q_0, X_1
Q_5	Q_4, X_0	Q_0, X_0
Q_6	Q_3, X_0	Q_5, X_1

表 2 は、入力{I_0, I_1, I_2, I_3}、状態{S_0, S_1, S_2, S_3, S_4}、出力{X_0, X_1} からなる不完全定義順序回路の状態遷移表および出力表である。表 2 中の＊はドントケアである。以下の(3)と(4)の問いに答えよ。なお、状態 Q と Q' が両立的であるとは、Q と Q' の両方に入力可能などのような入力系列に対しても、その出力系列が等しく、かつ両立的な状態に遷移することをいう。両立的な状態対を両立対と呼ぶ。

表 2 不完全定義順序回路の状態遷移表と出力表

	入力			
現状態	I_0	I_1	I_2	I_3
S_0	*, *	S_1, X_1	S_2, X_1	S_3, X_1
S_1	S_4, X_0	S_2, X_0	*, *	*, *
S_2	*, *	S_3, X_0	S_0, X_1	S_4, X_1
S_3	*, *	*, *	S_3, X_1	*, *
S_4	S_2, X_0	S_3, X_0	S_2, X_0	*, *

(3) 表 2 に対応する不完全定義順序回路における状態の両立対をすべて示せ。両立対の導出過程も明示せよ。

(4) 上記(3)で導出した両立対に基づいて、表 2 に対応する不完全定義順序回路の状態数を最小化し、最小化後の状態遷移表と出力表を示せ。最小化の過程と最小化前後の状態の対応関係を明示せよ。

Translation of technical terms

完全定義順序回路	completely specified sequential circuit
状態遷移表	state transition table
出力表	output table
等価	equivalent
同値類	equivalence class
不完全定義順序回路	incompletely specified sequential circuit
ドントケア	don't care
両立的	compatible
状態対	state pair
両立対	compatible pair

問6

算術論理演算命令，ロード命令，ストア命令，条件分岐命令を持つ5段のパイプライン処理プロセッサ（以下，プロセッサA）を考える．

算術論理演算命令の例として加算命令を示す．加算命令は2種類あり，レジスタ加算命令（add）は，レジスタRsとレジスタRtの保持する値を加算し，レジスタRdに代入する．即値加算命令（addi）は，レジスタRsの保持する内容にimmの値（即値）を加算し，レジスタRdに代入する．他の算術論理演算命令もこの2タイプとなる．

- add Rd, Rs, Rt
- addi Rd, Rs, imm

ロード命令（lw）とストア命令（sw）は，レジスタRsが保持する値にimmの値（即値）を加えたアドレスを対象とする．ロード命令はデータメモリからデータを読み込んでレジスタRdに代入する．ストア命令はレジスタRtの保持する値をデータメモリに書き込む．

- lw Rd, imm(Rs)
- sw Rt, imm(Rs)

条件分岐命令（bne）はレジスタRsとレジスタRtの保持する値が異なる場合，LABELのアドレスに分岐する．

- bne Rs, Rt, LABEL

プロセッサAにおける各パイプラインステージでの処理内容を表1に示す．各パイプラインステージは1クロックサイクルで実行する．

表 1: プロセッサAの各パイプラインステージにおける処理内容

ステージ	算術論理演算命令	ロード命令	ストア命令	条件分岐命令
IF	命令の読込み			
ID	命令の解釈，レジスタからの読込み			
EX	算術論理演算	アドレス計算		分岐先アドレス計算 分岐条件の判定
MEM	-	データの読込み	データの書込み	PCの変更
WB	レジスタへの書込み		-	-

プロセッサAのデータパスの概略図を図1に示す．プロセッサAでは，分岐先アドレス計算用と，算術論理演算/アドレス計算/分岐条件の判定用のALU（Arithmetic Logic Unit）が独立している．各ALUは2個の入力があり，後者のALUの片方の入力（ALU.Rs）は，各命令でRsとして指定したレジスタの値が入力される．また，命令メモリとデータメモリについても独立しており，両者に同時に1クロックサイクルでアクセスできる．レジスタはr0からr15までの16本あり，r0からは常に0を読み込むことができ，書込みは無効である．レジスタへの書込みは，クロックサイクルの前半に完了し，読出しについては，クロックサイクルの後半から開始できるものとする．なお，被ロード・データ・ハザードに対しては，IDステージで検出しストールすることにより，ハザードを回避する機構が組み込まれているとする．

図1において，IF/ID，ID/EX，EX/MEM，MEM/WBはパイプラインレジスタである．各パイプラインレジスタが保持する内容の一部を以下に示す．() 内は，内容が確定するステージを示す．ここで，IF/IDが保持するRegIDRsの内容はIF/ID.RegIDRsと表現する．

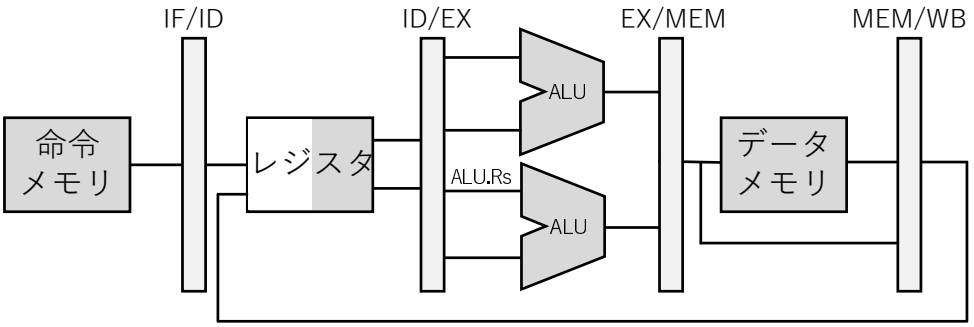

図 1: プロセッサ A のデータパスの概略図

- RegIDRs：各命令の Rs として指定したレジスタの番号（IF）
- RegIDRt：各命令の Rt として指定したレジスタの番号．使用しない命令では 0（IF）
- RegIDRd：各命令の Rd として指定したレジスタの番号．使用しない命令では 0（IF）
- RegRd：WB ステージでレジスタに書き込む値（EX または MEM）
- MemRead：1 なら MEM ステージでデータメモリを読込む（ID）
- RegWrite：1 なら WB ステージでレジスタに書込む（ID）

プロセッサ A に関して以下の問いに答えよ．

(1) EX ステージにおけるデータ・ハザードを解決するためのフォワーディング機構を検討する．ALU.Rs の入力として EX/MEM.RegRd を使用する（フォワードする）と判断するためのハザード検出条件は次のように表現できる．

- (EX/MEM.RegWrite == 1) and (not (EX/MEM.RegIDRd == 0))
  and (EX/MEM.RegIDRd == ID/EX.RegIDRs)

上記にならって，以下の条件を記述せよ．and や not の他に or も使用可能である．

- ALU.Rs の入力として MEM/WB.RegRd を使用する（フォワードする）と判断するためのハザード検出条件

(2) EX ステージにおけるデータ・ハザードを解決するためのフォワーディング機構を組み込んだプロセッサをプロセッサ B とする．プロセッサ B の ID ステージでの被ロード・データ・ハザードのハザード検出条件を（1）と同じフォーマットで示せ．また，被ロード・データ・ハザードを検出した場合，どれだけパイプラインをストールすればよいか．最小のクロックサイクル数を示し，その理由を説明せよ．

(3) プロセッサ B に対して，1 ビット分岐予測(ぶんきよそく)回路を加えたプロセッサ (プロセッサ B1) と，2 ビット分岐予測回路を加えたプロセッサ (プロセッサ B2) がある．それぞれの分岐予測回路の状態遷移図(じょうたいせんいず)を図 2 に示す．分岐予測回路は分岐毎に異なる状態を持つことが可能である．プログラム 1 とプログラム 2 のそれぞれに対して，1 行目から開始して最終行の分岐が成立せずプログラムが終了するまでプロセッサ B1 とプロセッサ B2 上で実行した場合，予測の精度はどちらのプロセッサが高いかその理由と共に書け．精度が同じ場合は「同じ」と書き，その理由を説明せよ．なお，#以降は各行の命令を説明したコメントであり，即値は 10 進数(しんすう)で記載している．

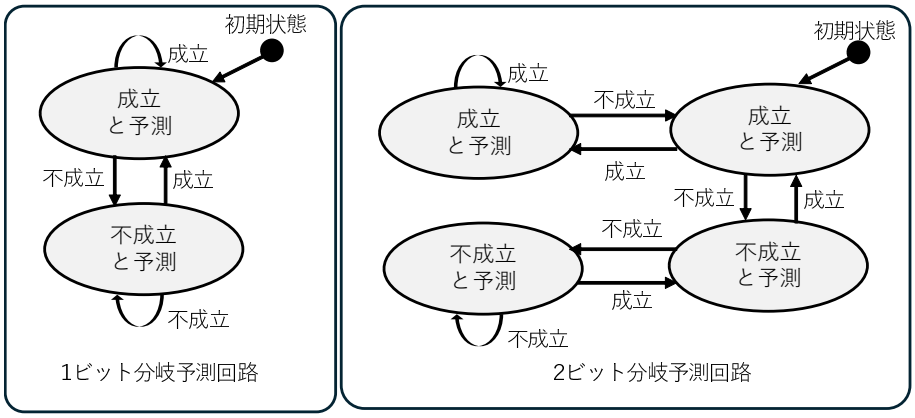

図 2: 1 ビット分岐予測回路と 2 ビット分岐予測回路の状態遷移図

プログラム 1

```
1     addi   r8, r0, 10      # r8 = 0 + 10
2     addi   r5, r0,  0      # r5 = 0 + 0
3  L1:add    r1, r1, r3      # r1 = r1 + r3
4     addi   r5, r5, 1       # r5 = r5 + 1
5     bne    r5, r8, L1      # r5とr8が等しくなければL1に分岐
```

プログラム 2

```
1     addi   r8, r0, 10      # r8 = 0 + 10
2     addi   r5, r0,  0      # r5 = 0 + 0
3  L1:add    r1, r1, r3      # r1 = r1 + r3
4     addi   r7, r0, 10      # r7 = 0 + 10
5     addi   r6, r0,  0      # r6 = 0 + 0
6  L2:add    r4, r4, r1      # r4 = r4 + r1
7     addi   r6, r6, 1       # r6 = r6 + 1
8     bne    r6, r7, L2      # r6とr7が等しくなければL2に分岐
9     addi   r5, r5, 1       # r5 = r5 + 1
10    bne    r5, r8, L1      # r5とr8が等しくなければL1に分岐
```

Translation of technical terms

算術論理演算	arithmetic logic operation
命令	instruction
ロード	load
ストア	store
条件分岐	conditional branch
パイプライン処理	pipeline processing
プロセッサ	processor
加算	addition
レジスタ	register
即値	immediate
アドレス	address
データメモリ	data memory
パイプラインステージ	pipeline stage
クロックサイクル	clock cycle
データパス	data path
命令メモリ	instruction memory
被ロード・データ・ハザード	load-use data hazard
パイプラインレジスタ	pipeline register
データ・ハザード	data hazard
ストール	stall
フォワーディング	forwarding
分岐予測	branch prediction
状態遷移図	state transition diagram
10 進数	decimal number

問 7

オペレーティングシステムにおける仮想メモリをページング方式で実現することについて，次の問いに答えよ。なお、仮想アドレスは 16 ビットで、そのうち上位 4 ビットで仮想ページ番号を、下位 12 ビットでページ内アドレスを示すものとする。

(1) 仮想メモリによって得られる利点を 2 つ挙げ、それぞれ説明せよ。
(2) 表 1 のアドレス変換表を使用する。存在ビット、参照ビット、変更ビットのそれぞれの意味を説明せよ。

表 1　アドレス変換表

仮想ページ番号	存在ビット	物理ページ番号	参照ビット	変更ビット
0x0	1	0x1	0	0
0x1	1	0x8	1	1
0x2	0	0xE	1	0
0x3	0	0xC	1	0
0x4	0	0xA	0	0
0x5	1	0x8	1	1

(3) 表 1 のアドレス変換表を用いて、次の仮想アドレスをアドレス変換した結果を回答せよ。アドレス変換ができる場合は、物理アドレスを 16 進数で表記せよ。アドレス変換できない場合は、「変換できない」と答えよ。
   （a）0x03E0
   （b）0x3500
   （c）0x5001
(4) ページ置き換えアルゴリズムである LRU について説明し、それが有効であると考えられている理由を説明せよ。
(5) 変更ビットが存在しない場合、オペレーティングシステムの処理はどのように変わるかを説明せよ。
(6) 仮想メモリにおけるスラッシングについて説明し、その問題点を説明せよ。

Translation of technical terms

仮想メモリ	virtual memory
ページング	paging
アドレス変換表	address translation table
存在ビット	present bit
参照ビット	reference bit
変更ビット	dirty bit
仮想アドレス	virtual address
物理アドレス	physical address
ページ置き換えアルゴリズム	page replacement algorithm
スラッシング	thrashing

問 8

プログラム P は与えられた文字列と値の対を保持，参照する C言語プログラムである．プログラム P に関する以下の問いに答えよ．

(1) プログラム P を実行した時に 86 行目の printf 関数により出力される値を答えよ．
(2) プログラム P を実行した時に 87 行目の printf 関数により出力される値を答えよ．
(3) プログラム P を実行する間に 46 行目の strcmp 関数が何回実行されるかを答えよ．
(4) プログラム P を実行する間に 68 行目の strcmp 関数が何回実行されるかを答えよ．
(5) プログラム Q に示す関数 delete_key をプログラム P の関数定義に加えることで，引数key で指定した文字列と値の対を削除できるようにしたい．プログラム Q の空欄①, ②を埋めよ．
(6) プログラム P を実行する間の 46 行目と 68 行目の strcmp 関数の実行回数を減らしたい．17 行目の hash 関数をどのように変更すればよいか答えよ．

なお，文字列は ASCII コードとして保持されるものとし，文字と ASCII コードの値の対応は以下の表に示すものとする．

文字	値
A	65
B	66

プログラム P, Q で呼び出しているライブラリ関数の仕様は以下のとおりである．

- char* strcpy(char *dest, char *src)
  引数 src が指す文字列を引数 dest が指す領域にコピーする．
- int strcmp(char *str1, char *str2)
  引数 str1 と引数 str2 が指す文字列が同一の文字列であれば，0 を返す．同一の文字列でなければ，0 以外の値を返す．
- int strlen(char *str)
  引数 str が指す文字列の文字数を返す．

Translation of technical terms

プログラム	program	コード	code
文字列	string	ライブラリ	library
C 言語	C programming language	指す	point
関数	function	領域	area
関数定義	function definition	コピー	copy
引数	argument		

プログラム P

```
1    #include <stdio.h>
2    #include <stdlib.h>
3    #include <string.h>
4
5    #define TABLE_SIZE 20
6
7    typedef struct node {
8        char *key;
9        int value;
10       struct node *next;
11   } node;
12
13   typedef struct hash_table {
14       node *nodes[TABLE_SIZE];
15   } hash_table;
16
17   unsigned int hash(char *key) {
18       unsigned int hash_value;
19       int i, l;
20       hash_value = 0;
21       l = strlen(key);
22       for (i = 0; i < l; i++) {
23           hash_value = hash_value + (unsigned int)key[i];
24       }
25       return hash_value % TABLE_SIZE;
26   }
27
28   hash_table* create_hash_table() {
29       int i;
30       hash_table *table;
31
32       table = malloc(sizeof(hash_table));
33       for (i = 0; i < TABLE_SIZE; i++) {
34           table->nodes[i] = NULL;
35       }
36       return table;
```

```
}

void put(hash_table *table, char *key, int value) {
    unsigned int index;
    node *current, *new_node;
    index = hash(key);
    current = table->nodes[index];

    while (current != NULL) {
        if (strcmp(current->key, key) == 0) {
            current->value = value;
            return;
        }
        current = current->next;
    }

    new_node = malloc(sizeof(node));

    new_node->key = malloc(strlen(key) + 1);
    strcpy(new_node->key, key);
    new_node->value = value;
    new_node->next = table->nodes[index];
    table->nodes[index] = new_node;
}

int get(hash_table *table, char *key) {
    unsigned int index;
    node *current;
    index = hash(key);
    current = table->nodes[index];
    while (current != NULL) {
        if (strcmp(current->key, key) == 0) return current->value;
        current = current->next;
    }
    return -1;
}

void main() {
```

```
75      hash_table *kv_store;
76      kv_store = create_hash_table();
77
78      put(kv_store, "AAA", 10);
79      put(kv_store, "AAB", 100);
80      put(kv_store, "ABA", 1000);
81      put(kv_store, "BAA", 10000);
82      put(kv_store, "ABA", 200);
83      put(kv_store, "ABB", 300);
84
85      printf("%d¥n", get(kv_store, "AAB"));
86      printf("%d¥n", get(kv_store, "ABA"));
87      printf("%d¥n", get(kv_store, "BBB"));
88  }
```

プログラムQ

```
1     void delete_key(hash_table *table, char *key) {
2         unsigned int index;
3         node *current, **prev;
4
5         index = hash(key);
6         current = table->nodes[index];
7         prev = [          ①          ];
8
9         while (current != NULL) {
10            if (strcmp(current->key, key) == 0) {
11                *prev = [          ②          ];
12
13                free(current->key);
14                free(current);
15                return;
16            }
17            prev = &(current->next);
18            current = current->next;
19        }
20    }
```