

# コンピュータのしくみ

吉川 雅弥 共著  
泉 知論

コロナ社

## まえがき

我々の身の回りには、マイコンをはじめとしてさまざまなコンピュータがあり、日々の生活においてそれらのコンピュータを使っている。「どのようなものにコンピュータが使われているか?」という質問には、多くの人が即答できると思うが、「コンピュータはどのように動作するのか?」という質問には、答えに困る人が出てくると思う。そこで本書は、大学生や若手技術者を対象に、コンピュータのしくみを説明したものである。1章ではコンピュータの構成や動作原理について概説し、2章ではコンピュータの中での情報の表現方法について詳細に説明している。また、3章ではコンピュータのさまざまな機能を実現する論理演算について解説し、4章ではデータを保持する機構と、演算器や記憶装置の接続方法について説明している。また、5章では演算について、その原理とハードウェアでの構成方法について説明し、6章ではアセンブリ言語について説明している。このアセンブリ言語は、一般的に機種によって異なるが、本書では経済産業省の情報処理技術者試験の受験も考えて、当該試験で用いられる CASL II を使用している。最後に、7章ではコンピュータの中の演算器などをどのように制御するかについて解説している。

すべての章を勉強することで、「コンピュータはこのように動作する」と、自信を持って答えられるようになることを考える。大学や高专での教科書、若手技術者の参考書として利用していただければ幸甚である。本書では、著者らの大学におけるこれまでの講義での学生の理解度を鑑みて、動作原理などを解説している。重要な部分については、例題を付けているので、ぜひ解いてもらいたい。また、発展的な内容については「コーヒープレイク」に書いてあるので、より掘り下げて学習したい場合には、「コーヒープレイク」も熟読してもらいたい。

本書の執筆にあたっては、数多くの関連する専門書を参考にさせていただい

た。主要なものについては、関連する内容ごとに巻末に挙げている。漏れているものもあるかと思うが、その点をご容赦願いたい。

本書の出版にあたって、コロナ社の関係各位には叱咤激励も含めていろいろとお世話になり感謝したい。また、本書での図の作成にあたり、野崎佑典氏には全面的に協力していただいた。ここに深く感謝申し上げる。

2016年12月

吉川雅弥，泉 知論

# 目 次

## 1. コンピュータの基礎

1.1 コンピュータの構成 .....	1
1.2 命 令 セ ッ ト .....	2
1.3 プロセッサの基本動作 .....	5
1.4 アドレッシング .....	9
1.5 記 憶 装 置 .....	14
1.6 接 続 方 法 .....	15
1.7 性 能 評 価 .....	16
章 末 問 題 .....	20

## 2. 情報 の 表 現

2.1 位取り記数法 .....	22
2.2 数 の 接 頭 語 .....	23
2.3 ビットとバイト .....	24
2.4 基 数 変 換 .....	26
2.5 負 数 表 現 .....	30
2.6 負 数 の 演 算 .....	33
2.7 コ ー ド .....	37
2.8 実 数 表 現 .....	42
2.9 浮動小数点数の演算 .....	47

2.10 数値演算の誤差	49
章 末 問 題	51

### 3. 論理の世界

3.1 集 合	53
3.2 命 題	56
3.3 論理での演算	62
3.4 論理関数	63
3.5 標準形	65
3.6 論理関数の表現方法	68
章 末 問 題	71

### 4. 記憶と接続

4.1 レジスタ	72
4.1.1 SR ラッチ	74
4.1.2 D ラッチ	76
4.1.3 マスタスレーブ型 D フリップフロップ	78
4.1.4 セットアップタイムとホールドタイム	79
4.2 接 続	81
4.2.1 セレクタ	82
4.2.2 バス	83
4.3 メモリ	87
4.3.1 メモリの種類と特徴	87
4.3.2 プロセッサとメモリ	89
4.3.3 アドレス空間と共有	92
4.3.4 メモリマップド I/O	94
4.3.5 DMA	95

4.3.6 キャッシュ	97
4.3.7 インタリーブ	99
章 末 問 題	99

## 5. 演 算

5.1 加 算 器	102
5.2 加 減 算 器	106
5.3 フ ラ グ	108
5.4 シフトとローテート	110
5.5 ALU	115
5.6 乗 算	116
章 末 問 題	123

## 6. コンピュータの言葉

6.1 CASL II と COMET II	126
6.2 CASL II の命令	130
6.2.1 データ転送命令	131
6.2.2 算術・論理演算命令	133
6.2.3 比 較 命 令	136
6.2.4 シフト命令	137
6.2.5 分 岐 命 令	138
6.3 プログラムの記述方法	139
6.4 基本的なプログラムの例	142
6.5 CASL II と機械語	148
6.6 機械語と主記憶	152
章 末 問 題	155

## 7. 制 御

7.1 有限状態機械 .....	157
7.2 制御回路とデータパス .....	159
7.3 高度な制御 .....	162
7.4 割込みと例外 .....	167
7.4.1 割 込 み .....	167
7.4.2 例 外 .....	173
章 末 問 題 .....	174
付 録 —— COMET II と CASL II の仕様 —— .....	175
引用・参考文献 .....	184
索 引 .....	187

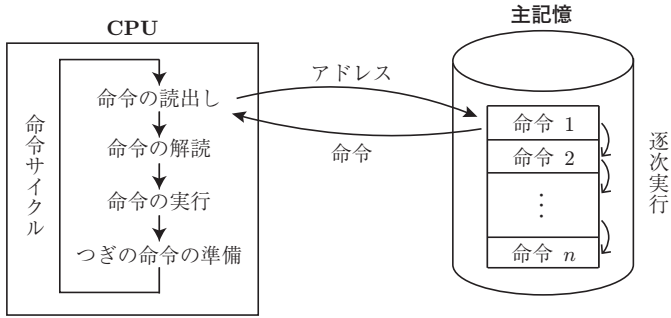


図 1.5 命令の実行方法

表 1.3 命令サイクル

処 理	ステップ	内 容
命令の取得	(1)	主記憶からの命令の読み出し
	(2)	命令の解釈
オペランドの処理	(3)	アドレスの計算
	(4)	オペランドの読み出し
	(5)	命令の実行
	(6)	結果の格納
つぎの命令の準備	(7)	プログラムカウンタの更新

命令をデコーダを使用して解釈する。デコーダとは、あらかじめ用意された命令に対して、制御信号を選択する回路である。この選択された信号によって制御装置が当該命令に対応する処理を実行する。

つぎに、(3)~(6)のオペランドの処理では、オペランドの読み出しや結果の書込みに必要なアドレスを計算する。計算したアドレスを利用して、主記憶またはレジスタからオペランドを読み出す。その後、解釈した命令の内容（例えば、加減算、乗除算などの四則演算および論理演算）に従い、データに対する演算や加工を行い、その結果を主記憶またはレジスタに格納する。最後に、つぎの命令のアドレスを計算し、プログラムカウンタに設定する。

基本的な動作は、(1)~(7)の繰返して、実際のコンピュータでは、いくつかのステップを同時に実行する。これらの処理ステップの制御を行う制御装置には、二つの構成方式がある。一つはマイクロプログラム制御方式 (microprogrammed control) で、もう一つは、布線論理制御方式 (wired logic control) である。



- さい。
- (3)  $S = \sum_{n=1}^{\infty} \frac{1}{n!}$  で生じる誤差は、どのような誤差か答えなさい。

**【解答】**

- (1) 有効桁数が失われる桁落ちの誤差が生じる可能性がある。
- (2)  $x \gg y$  では、 $x + y = x$  となる。すなわち、 $y$  は、 $x$  と同じ扱いになり、無視される。したがって、このような誤差は情報落ちである。
- (3)  $1/n!$  が循環小数になる場合があり、丸め誤差が生じる。さらに、無限に  $1/n!$  を加算することはできないため、有限回で処理を打ち切り、打ち切り誤差が生じる。 ◇

## 章 末 問 題

- 【1】 10 進数 672 を、2 進数と 16 進数に変換しなさい。
- 【2】 16 進数 E4A を、8 進数に変換しなさい。
- 【3】 10110010 が「1 の補数表現」の場合、いくつの値を示しているか 10 進数で答えなさい。また、2 の補数表現の場合についても答えなさい。
- 【4】 2 の補数表現の場合、1 の補数表現より表現できる数が多い。その理由を簡潔に説明しなさい。
- 【5】 1 の補数表現において、循環桁上りの処理をしないと、正しい演算結果が得られない理由について、簡潔に述べなさい。
- 【6】 10 進数 0.7 を、2 進数に変換しなさい。
- 【7】 9 進数 12.34 を、3 進数に変換しなさい。
- 【8】 つぎのビットパターンが IEEE 754 の単精度で表現されている場合、いくつの値を示しているか 10 進数で答えなさい。
- (1) 01010010110110000000000000000000
- (2) 11000111101110100000000000000000
- 【9】 10 進数 6.125 と 10 進数 5.75 を、2 進数の浮動小数点方式に変換して、加算を行いなさい。ただし、仮数の桁数を 4 桁として、丸めの方法は、零捨一入とする。
- 【10】 10 進数 -2.25 と 10 進数 0.4375 を、2 進数の浮動小数点方式に変換して、乗算を行いなさい。ただし、仮数の桁数を 4 桁として、丸めの方法は、零捨一入

の多数決を取る関数となり、それぞれ論理式は式 (5.1), 式 (5.2) になる。論理回路を図 5.4 に示す。

$$s_i = a_i \oplus b_i \oplus c_i = a_i \bar{b}_i \bar{c}_i + \bar{a}_i b_i \bar{c}_i + \bar{a}_i \bar{b}_i c_i + a_i b_i c_i \quad (5.1)$$

$$c_{i+1} = a_i b_i + b_i c_i + c_i a_i \quad (5.2)$$

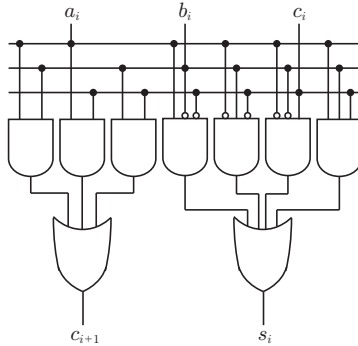


図 5.4 1 ビット全加算器の論理回路

1 ビットの全加算器を並べ、桁上げを下位から上位に 1 桁ずつつなぐことで多ビットの加算器が実現できる。図 5.5 に 4 ビット加算器の構成を例示する。このような構成の加算器を桁上げ伝搬加算器 (ripple carry adder, RCA, ripple は「さざ波」の意) と呼ぶ。

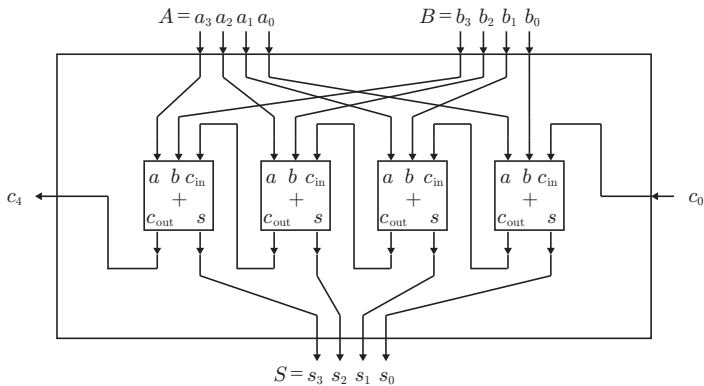


図 5.5 4 ビットの桁上げ伝搬加算器

1 :	LD	GR1, NUM	: ラベル NUM のアドレスの内容を GR1 へ
2 :	ADDA	GR1, GR2	: GR1 と GR2 を加算

図 6.28 機械語変換用の例題

つぎに、2 行目の ADDA 命令は表 6.8 より 1 語構成の命令となる。15 $\cdots$ 8 (8 ビット) は、00100100 で、7 $\cdots$ 4 (4 ビット) と 3 $\cdots$ 0 (4 ビット) は、それぞれ GR1 と GR2 に対応するため、0001 と 0010 となる。16 進数表記をすれば 2412 となる。このようにアセンブリ言語から機械語へ変換することをアセンブルするという。

インデックスレジスタに GR0 が使用できない理由は、使用しない場合を 0 としているため、もし GR0 をインデックスレジスタとして使用した場合、未使用の 0 か GR0 かの違いを区別することができない。

**例題 6.3** 機械語への変換について、つぎの問いに答えなさい。

- (1) 図 6.29 はあるプログラムの一部である。表 6.8 を用いて、図 6.29 を機械語に変換しなさい。ただし、ラベル N1, N2, N3 の示すアドレスをそれぞれ 000Ah, 000Bh, 000Ch とする。

1 :	LD	GR1, N1
2 :	LD	GR2, N2
3 :	ADDA	GR1, GR2
4 :	ST	GR1, N3
5 :	RET	

図 6.29 アセンブリ言語から機械語への練習問題

- (2) つぎの機械語を CASL II のアセンブリ言語に変換しなさい。ただし、各命令は、16 進数で表記されているものとする。
- (a) 3456  
 (b) 52200008  
 (c) 10730004

することができない。

(3) パイプライン処理による性能向上を実現するためには、十分な数の命令が必要である。

(4) パイプライン処理によって、レイテンシを改善することができる。

### 【解答】

(1) 間違い。理想的な状態で  $n$  倍になる。そのため、現実的には数が多ければ  $n$  倍に近づく。

(2) 間違い。すべてのステージの処理時間が同じほうが、性能が向上しやすいのであって、同じでなければパイプライン化できないわけではない。

(3) 正しい。

(4) 間違い。パイプライン処理で改善するのは、スループット。

したがって、答えは、(3)。

◇

パイプライン処理は、図 7.5 に示したように、ある命令を実行しているつぎのクロックサイクルで、つぎの命令を実行する。しかし、これを実行できない場合がある。これをハザードと呼び、構造ハザード、データハザード、制御ハザードの三つがある。

まず、構造ハザードは、並列に実行されている命令に、演算器、レジスタ、メモリなどのハードウェアのリソースが不足している場合に起きる。図 7.6 に構造ハザードの例を示す。図の例では、メモリへのデータの書き込み（または読出し）するためのメモリへのアクセス (MEM) と、命令をフェッチするためのメモリへのアクセス (IF) で競合が起きている。

つぎに、データハザードは、先行命令の演算結果を後続の命令が利用する場合

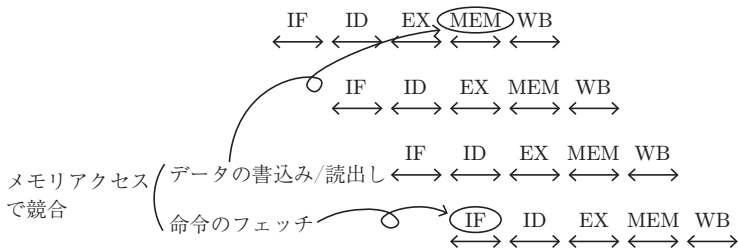


図 7.6 メモリへのアクセスでの構造ハザード

# 索引

	間接アドレス指定	11			
<b>【あ】</b>	<b>【き】</b>		<b>【さ】</b>		
アセンブラ	3	記憶階層	14	最小項	64
アセンブラ言語	3	機械語	2	最大項	64
アセンブラ命令	130	疑似命令	130	算術シフト	111
アセンブリ言語	3	基底アドレス	10		
アドレス	6	キャッシュ	97	<b>【し】</b>	
アドレス空間	92	キュー	129	自己相対アドレス指定	12
アドレスデコーダ	93	共通部分集合	54	指数部	43
あふれフラグ	108			実効アドレス	9
アンドフロー	44	<b>【く】</b>		指標レジスタ	128
		空集合	53	シフト	110
<b>【い】</b>		位取り記数法	22	シフトレジスタ	113
インタリーブ	99	クリティカルパス	161	シャノンの展開定理	64
インデックス・アドレス指定	11	グレイコード	39	集合	53
		クロスバススイッチ	86	主加法標準形	65
<b>【う】</b>		クロック周期	16	主記憶	6
打ち切り誤差	50	クロック周波数	16	主乗法標準形	65
		<b>【け】</b>		循環桁上り	34
<b>【え】</b>		桁上げ先見加算器	105	乗算器	116
演算ユニット	6	桁上げ伝搬加算器	104	状態遷移図	157
		桁上げフラグ	108	状態遷移表	157
<b>【お】</b>		桁落ち	50	情報落ち	50
応答時間	16	<b>【こ】</b>		シリアル	113
オーバフロー	44	語	2	シリアル-パラレル変換	113
オペコード	3	恒偽命題	59	真部分集合	54
オペランド	3	恒真命題	59	真理値	57
		構造ハザード	164	真理値表	57
<b>【か】</b>		語長	2	<b>【す】</b>	
外延的記法	53	固定小数点	42	スタック	129
科学記数法	44			スタックポインタ	128
加減算器	107			ステージ	162
加算器	102			ストリーム型	
仮数部	43			インタフェース	96

ストール	165			ブルダウン	86
スリーステートバッファ	84			プログラムカウンタ	6
スループット	18, 163			プログラムレジスタ	6, 128
<b>【せ】</b>		<b>【な】</b>		<b>【へ】</b>	
制御ハザード	164	内包的記法	53	平均故障間隔	19
積集合	54	ニモニック	3	平均修理時間	19
絶対アドレス	9	<b>【の】</b>		ベン図	56
絶対値表現	30	ノイマン型コンピュータ	1	ベンチマーク	20
セットアップタイム	80	ノイマン・ボトルネック	15	<b>【ほ】</b>	
セレクト	81, 82	<b>【は】</b>		補集合	53
零フラグ	108	倍精度	44	補数表現	30
全加算器	103	バイト	24	ポップ	129
<b>【そ】</b>		パイプライン処理	162	ポーリング	168
相対アドレス	10	バス	15, 82, 83	ホールドタイム	80
即 値	10, 145	バスアービタ	97	<b>【ま】</b>	
ソフトウェア割込み	172	バスコントローラ	97	マイクロプログラム制御	7
<b>【た】</b>		バブル	166	マクロ命令	130
多重割込み	171	パラレル	113	交わり	54
多相同期回路	77	パラレル-シリアル変換	113	マシン語	2
単精度	44	バレルシフト	114	マスク ROM	88
<b>【ち】</b>		汎用レジスタ	128	マスタスレーブ	78
中央処理装置	2	<b>【ひ】</b>		マルチサイクル方式	162
直積集合	54	ビジーウエイト	167	マルチプレクサ	82
直接アドレス指定	10	ビット	24	丸め誤差	50
<b>【て】</b>		非同期リセット信号	80	<b>【み】</b>	
デコーダ	7	<b>【ふ】</b>		ミーリ型	158
データハザード	164	フィールド	148	<b>【む】</b>	
データバス	159	符号フラグ	108	ムーア型	158
<b>【と】</b>		布線論理制御	7	無限集合	53
投機実行	166	プッシュ	129	矛盾	59
同期ロード信号	80	浮動小数点	42	結 び	54
トークンリング	86	部分集合	54	<b>【め】</b>	
トートロジー	59	フラグ	108	命 題	56
ド・モルガンの法則	60	フラグ付きローテート	111	命令形式	2
トライステートバッファ	84	フラグレジスタ	128	命令サイクル	6
		フラッシュメモリ	89	命令セット	4
		フリップフロップ	73	命令レジスタ	6
		プルアップ	86		
		プール代数	63		

メモリ型インタフェース	96	ライトバック	98		
メモリ空間	92			<b>【ろ】</b>	
メモリバス	89	<b>【り】</b>		ローテート	110
メモリマップ	93	リタイミング	161	論理関数	63
メモリマップドレジスタ	95	リテラル	64	論理ゲート	68
メモリマップド I/O	94	リングオシレータ	75	論理シフト	111
<b>【ゆ】</b>		<b>【れ】</b>		<b>【わ】</b>	
有限集合	53	例 外	173	和集合	54
有限状態機械	157	レイテンシ	18, 162	割込み	168
<b>【よ】</b>		レジスタ	5, 72	割込みイネーブル	170
要 素	53	レジスタ・アドレス指定	11	割込み復帰命令	171
<b>【ら】</b>		レジスタ型		割込みフラグ	168
ライトスルー	97	インタフェース	95	割込みベクタ	171
				割込みマスク	170



		CLA	105	empty set	53
<b>【A】</b>		clock cycle per instruction		EPROM	89
absolute address	9	COMET II	126, 176	EUC	37
ALU	6, 115	complement	54	exception	173
AND	68	complex instruction		<b>【F】</b>	
arithmetic logic unit	6, 115	set computer	8	FF	73
ASCII	37	CPI	17	FIFO	129
assembler	3	CPU	2	finite state machine	157
assembly language	3	CPU 時間	16	flash memory	89
<b>【B】</b>		cross-bar switch	86	flipflop	73
barrel shifter	114	<b>【D】</b>		full adder	103
base address	10	D フリップフロップ	78	<b>【I】</b>	
BCD	39	D ラッチ	76, 77	IEEE 754	44
binary coded decimal	39	datapath	159	interleave	99
bit	24	direct memory access	97	intersection	54
bus	82	D-latch	76	IR	6
byte	24	DMA	97	ISR	169
<b>【C】</b>		DRAM	88	<b>【J】</b>	
cache	97	<b>【E】</b>		JIS 8 ビット	37
carry lookahead adder	105	EEPROM	89	join	54
CASL II	126, 179	effective address	9		
central processing unit	2	element	53		
CISC	8				

			OR	68	SR ラッチ	74
			overflow flag	128	SRAM	87
					SR-latch	74
		<b>[L]</b>			<b>[T]</b>	
least significant bit	24		parallel	113	three-state buffer	84
LIFO	129		PC	6	token ring	86
literal	64		pop	129	tri-state buffer	84
LSB	24		PR	6		
		<b>[M]</b>	PROM	88		
maxterm	64		proposition	56		
mean time between failures	19		push	129		
mean time to repair	19					
meet	54				<b>[U]</b>	
MFLOPS	18				union	54
microprogrammed control	7				UV-EPROM	89
million floating-point			<b>[R]</b>			
operations per second	18		RAM	87		
million instructions per			reduced instruction		<b>[V]</b>	
second	18		set computer	8	Venn diagram	56
minterm	64		register	72, 73		
MIPS	18		relative address	10	<b>[W]</b>	
mnemonic	3		resistor	73	wired logic control	7
most significant bit	24		ring oscillator	75	word	2
MSB	24		ripple carry adder	104		
MTBF	19		RISC	8	<b>[X]</b>	
MTTR	19		ROM	87	XOR	68
multiplexor	82		rotate	110		
MUX	82				<b>[Z]</b>	
					zero flag	128
			<b>[S]</b>		ZF	128
			selector	81	~~~~~	
			serial	113		
			set	53	<b>[数字]</b>	
			SF	128	1 の補数	31
			shift	110	2 の補数	31
			Shift-JIS	37	3 増しコード	39
			sign flag	128		
		<b>[O]</b>				
OF	128					
operand	3					
operation code	3					



— 著者略歴 —

吉川 雅弥 (よしかわ まさや)  
2001年 立命館大学大学院理工学研究科  
博士後期課程修了, 博士 (工学)  
1998年 立命館大学第一号助手  
2001年 立命館大学博士研究員  
2004年 立命館大学講師  
2007年 名城大学准教授  
2012年 名城大学教授  
現在に至る

泉 知論 (いずみ ともりの)  
1998年 東京工業大学大学院理工学研究科  
博士後期課程修了, 博士 (工学)  
1998年 京都大学助手  
2005年 立命館大学助教授  
2007年 立命館大学准教授  
2016年 立命館大学教授  
現在に至る

## コンピュータのしくみ

Mechanism of Computers

© Masaya Yoshikawa, Tomonori Izumi 2017

2017年 2月27日 初版第1刷発行



検印省略

著者 吉川 雅 弥  
泉 知 論  
発行者 株式会社 コロナ社  
代表者 牛来真也  
印刷所 三美印刷株式会社

112-0011 東京都文京区千石 4-46-10

発行所 株式会社 コロナ社

CORONA PUBLISHING CO., LTD.

Tokyo Japan

振替 00140-8-14844・電話(03)3941-3131(代)

ホームページ <http://www.coronasha.co.jp>

ISBN 978-4-339-02867-6 (新井) (製本:愛千製本所)

Printed in Japan



本書のコピー、スキャン、デジタル化等の無断複製・転載は著作権法上での例外を除き禁じられております。購入者以外の第三者による本書の電子データ化及び電子書籍化は、いかなる場合も認めておりません。

落丁・乱丁本はお取替えいたします