計算機システム I ・第 13 回

2022 年 7 月 13 日

今回の内容

前回紹介した「簡単な計算器」は、4 bit のデータ入力と 4 bit のデータ出力を持ち、4 通りの操作を実行可能なものでしたが、実行する操作は、クロック信号の立ち上がりごとに、毎回、2 bit の制御信号として入力してやらなければなりませんでした。今回は、実行したい一連の操作を計算機プログラムとして記憶しておき、そのプログラムに基づいて動作する、いわゆる一般の CPU (中央処理装置)[1]として働く論理回路を考えてみます。

13.1 単純な 4 bit CPU

今回紹介するのは、単純な 4 bit CPU の論理回路で、次の 14 種類の機械語命令を実行できます。

機械語命令のビットパターン								命令	動作
I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0		
0	0	0	0	0	0	0	0	NOP	何もしない
0	0	0	0	0	0	1	0	OUT A	A レジスタの値を *OUT* へ出力
0	0	0	0	0	1	0	1	CMP A, B	A−B の計算結果を CF へ反映
0	0	0	0	1	0	0	0	CPY B, A	A レジスタ の値を B へコピー
0	0	0	1	0	1	0	0	ADD B, A	A+B の計算結果を B へ書き込み
0	0	0	1	0	1	0	1	XSUB B, A	A−B の計算結果を B へ書き込み
0	0	0	1	1	0	0	0	IN B	*IN* の値を B レジスタへ書き込み
0	0	1	0	0	0	0	0	CPY A, B	B レジスタの値を A へコピー
0	0	1	0	1	0	0	0	SWP A, B	A レジスタ と B レジスタの値を交換
0	1	0	0	0	1	0	0	ADD A, B	A+B の計算結果を A へ書き込み
0	1	0	0	0	1	0	1	SUB A, B	A−B の計算結果を A へ書き込み
0	1	1	0	*data*				CPY A, *data*	定数 data を A レジスタへ書き込み
1	0	*address*						JPC *address*	CF=1 ならば address 番地にジャンプ
1	1	*address*						JMP *address*	常に address 番地にジャンプ

この表からも分かるように、この CPU は次のようなものとなっています。

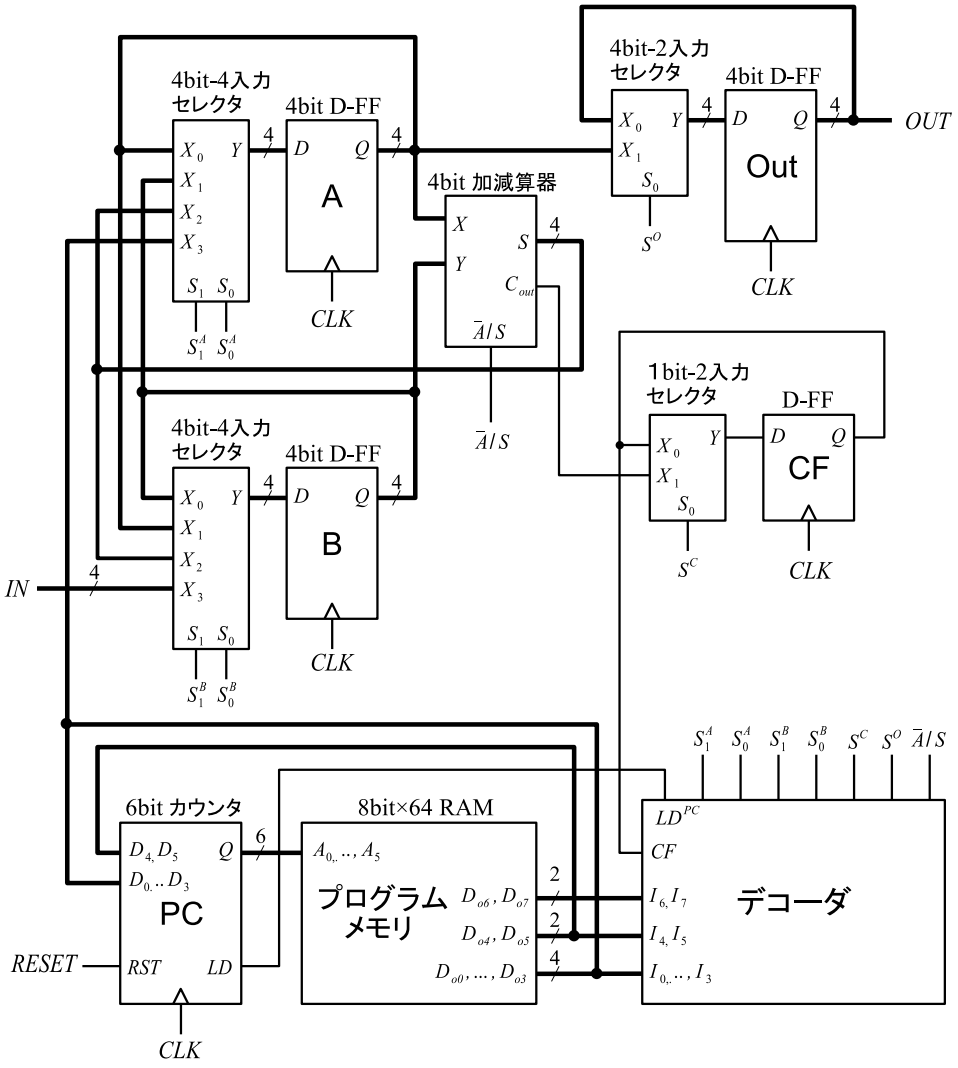
- 機械語命令はどれも 8 bit 長である。
- 64 B (バイト[2]) の容量のプログラムメモリが付属しており、そこに記憶された最大 64 個の機械語命令からなるプログラムを実行できる。
- プログラムメモリは機械語プログラムの記憶専用で、データを記憶することはできない。
- 2 つの 4 bit のレジスタ (A と B) を持つ。
- これら 2 つのレジスタ間で 4 bit の符号なし整数の加算と減算ができる。
- 加算や減算で桁溢れが起きたかどうかを示す CF (キャリーフラグ) を持っている。

[1] Central Processing Unit

[2] 1 B (バイト) = 8 bit (ビット)

- 無条件分岐命令と CF=1 のときのみ分岐する条件付き分岐命令を持つ。
- 4 bit の入力 *IN* と 4 bit の出力 *OUT* を持ち、機械語命令によって、*IN* の値をレジスタに取り込んだり、レジスタの値を *OUT* に出力したりすることができる。

4 bit CPU の回路図　次は、この CPU の回路図です。回路図中の $\overset{4}{\not\!/}$ や $\overset{6}{\not\!/}$ は、それぞれ 4bit や 6bit 分の接続 (4 本や 6 本の電線) を表していることに注意してください。この CPU には、第 6 回で紹介した加算器や減算器、前回紹介した 多 bit 長の D-フリップフロップ (D-FF) とセレクタが使われています。「PC (プログラムカウンタ)」は第 10 回で紹介したカウンタの一種ですが、「プログラムメモリ」や「デコーダ」とともに後程その詳細について説明します。

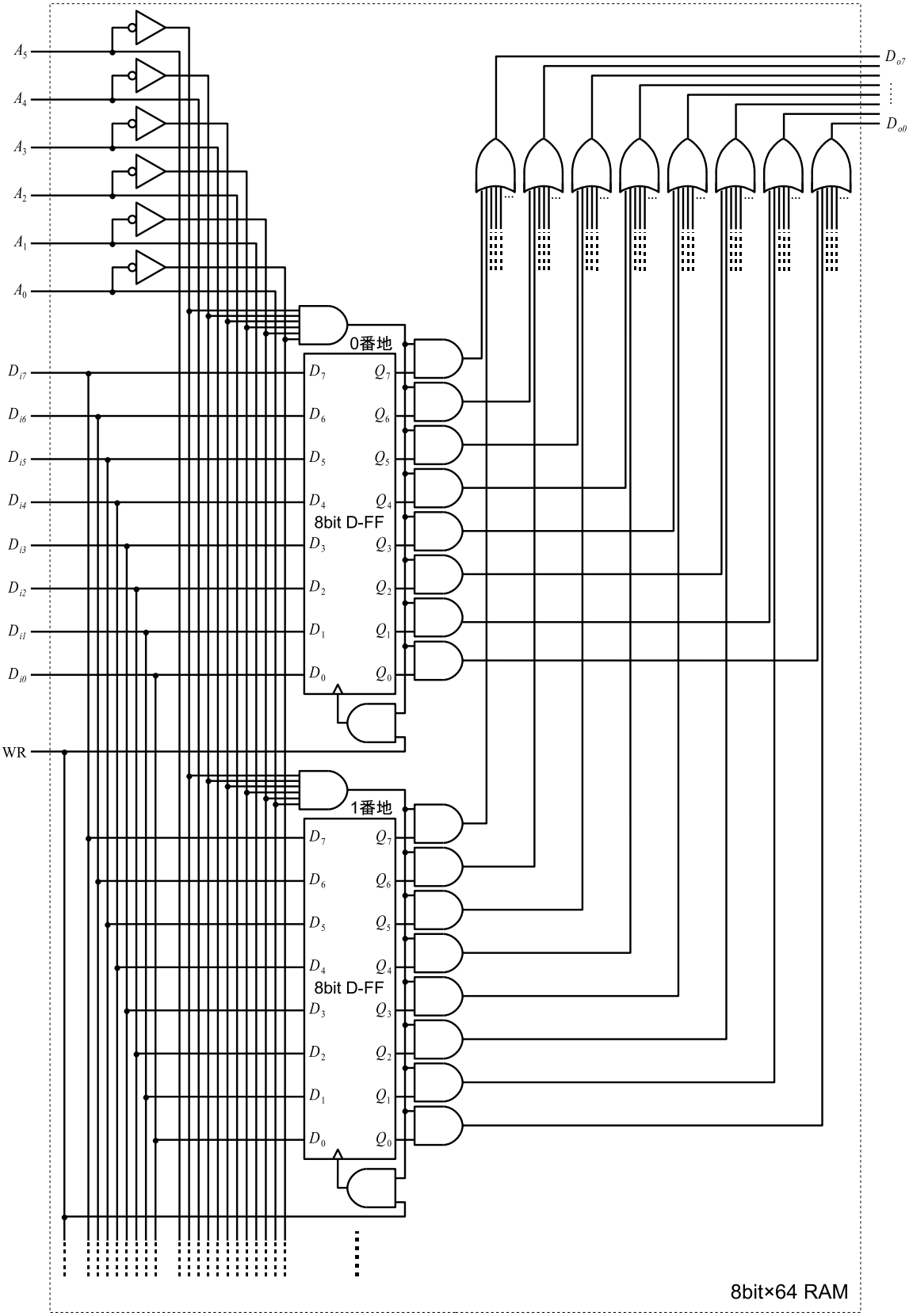


4 bit CPU の回路図

デコーダの出力 S_1^A, S_0^A, S_1^B, S_0^B, S^C, S^O は各セレクタの入力へ、LD^{PC} は PC の LD へ、$\overline{A}/S$ は加減算器へ接続します。この CPU を動作させるためには、適当な周波数のクロック信号を用意し、PC や各 D-FF (D-フリップフロップ) の *CLK* へ接続します。

あらかじめ、プログラムメモリ (8bit×64 RAM) に機械語プログラムを記憶しておき、*RESET* を 1 から 0 に変更すると、クロック信号に同期して、0 番地から順に機械語命令が実行されていきます。A レジスタ、B レジスタ、Out レジスタ、CF の初期値は不定となります。

プログラムメモリ　この CPU の回路図に現れている「プログラムメモリ」は、8 bit のデータを 64 個記憶できる順序論理回路です。この「プログラムメモリ」は、たとえば、8 bit の D-フリップフロップを 64 個使用した次のような回路として実現することができます。

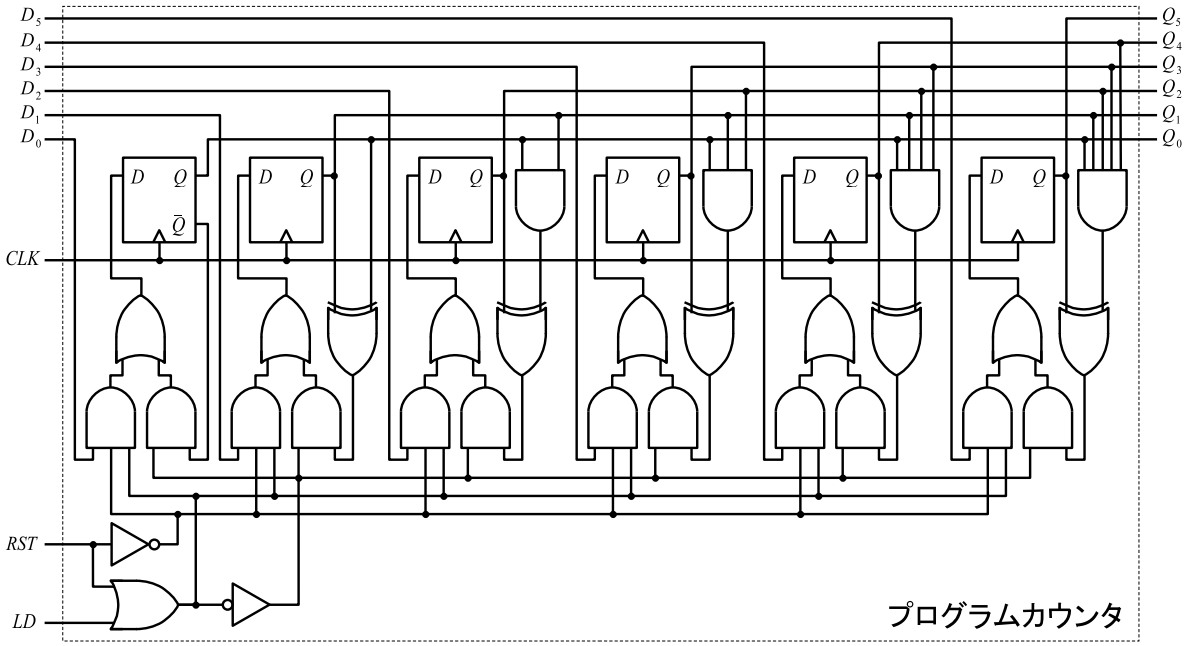


このプログラムメモリに機械語プログラムを記憶させるためには、まず、$A_5\,A_4\,A_3\,\ldots\,A_0$ の 6 bit

で番地 (アドレス) を指定し、$D_{i7}\,D_{i6}\,D_{i5}\,\ldots\,D_{i0}$ の 8 bit にその機械語命令のビットパターンを入力した上で、WR を 0 から 1 へ変更して、すぐに 0 に戻します[3]。これにより、指定した番地に指定した 8 bit のビットパターンが記憶されます。これを繰り返し、プログラムメモリ中に 64 個までの機械語命令を記憶しておくことができます。

このようにして記憶された機械語命令は、CPU の実行時には、PC (プログラムカウンタ) の 6 bit の出力 Q が $A_5\,A_4\,A_3\,\ldots\,A_0$ に入力されることで、プログラムメモリの出力 $D_{o7}\,D_{o6}\,D_{o5}\,\ldots\,D_{o0}$ に読み出されます。

プログラムカウンタ　PC (プログラムカウンタ) は、この CPU が次に実行すべき機械語命令の番地 (アドレス) を記憶する順序回路です。CPU は、クロック信号に同期して、この PC が指定した番地からプログラムメモリ中の機械語命令を 1 つ読み出して、その命令を実行しますが、これと同時に PC が記憶している番地は 1 進みます。プログラムメモリの番地 (アドレス) は 6 bit で表しますので、この CPU の PC は 6 bit のカウンタ[4]として動作することになります。ただし、機械語命令の中には、JMP 命令や JPC 命令にように、次に実行する命令の番地を変更する命令 (分岐命令) がありますので、PC は、単に番地を 1 進めるだけではなく、それを指定した値に設定する機能を持っていなければなりません。このような機能は、たとえば次のような論理回路で実現できます。



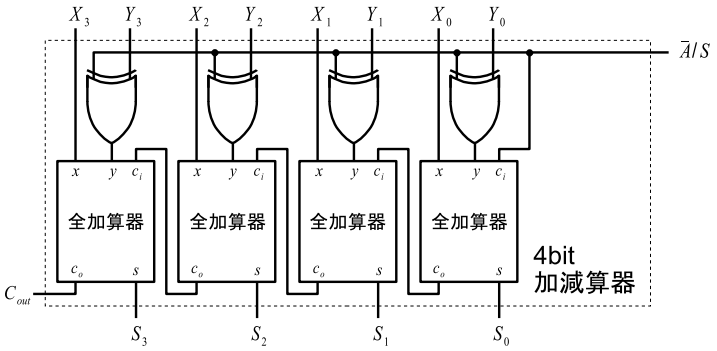
通常は、$RST = LD = 0$ としておきます。この状態の PC は、単なる 6 bit の (同期式) カウンタとして動作します。PC を特定の値に設定したい場合は、その値を $D_5\,D_4\,D_3\,\ldots\,D_0$ に入力し、LD を 1 にします。すると、次のクロック信号 (CLK) の立ち上がりで、PC はその値に設定されます。その後、$LD = 0$ に戻しておけば、その値から (クロック信号に同期して) カウントアップしていきます。同様に、(LD の代りに) RST を 1 にすれば、PC の値を 0 にリセットすることができます。

---

[3] WR は通常は 0 にしておきます。

[4] カウンタについては第 10 回の資料を参照してください。

加減算器　加減算器は、第 6 回で紹介した加算器と減算器を 1 つにまとめたもので、次のような論理回路です。入力 $\overline{A}/S$ を 0 にすれば加算器として、1 にすれば減算器として働く論理回路となっています。



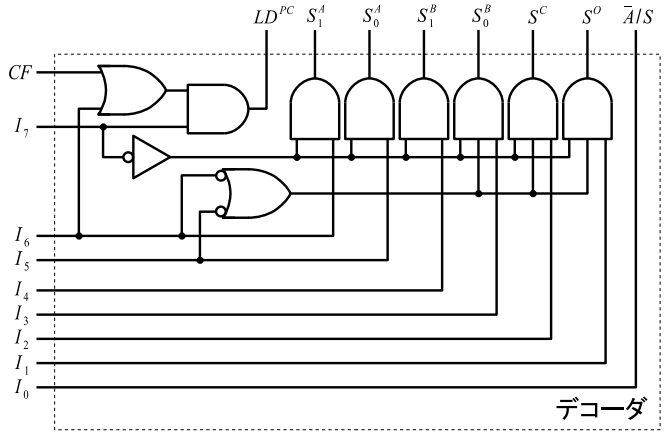
デコーダ　プログラムメモリ中に記憶されている機械語プログラムを固定して考えると、この CPU の状態は、A レジスタの値、B レジスタの値、Out レジスタの値、CF (キャリーフラグ) の値、PC の値で決まります。これらの値はすべてクロック信号の立ち上がりで変化しますが、それぞれの機械語命令に応じて、これらが次にどのような値に変化すべきかを指定することができれば、この CPU の動作を実現することができます。

A レジスタの次の値は、その入力側に設置されているセレクタの入力 $S_1^A\,S_0^A$ と加減算器の入力 $\overline{A}/S$ および機械語命令の下位 4 bit で決まりますし、B レジスタの次の値は、その入力を切り替えている $S_1^B\,S_0^B$ と加減算器の入力 $\overline{A}/S$ で決まります。また、CF (キャリーフラグ) と Out レジスタは、それぞれ S^C と S^O で、PC については、その入力 LD と $D_5\,D_4\,D_3\,\ldots\,D_0$ で決まります。これらのうち、機械語命令の下位 4 bit を A レジスタのセレクタに、下位 6 bit を PC の $D_5\,D_4\,D_3\,\ldots\,D_0$ へ、2 ページの回路図のように接続しておけば、結局、プログラムメモリから読み取った機械語命令の各ビットパターン $I_7\,I_6\,I_5\,\ldots\,I_0$ に対して、LD^{PC}, S_1^A, S_0^A, S_1^B, S_0^B, S^C, S^O, $\overline{A}/S$ の 8 つの真理値を次の表のように設定すればよいことが分かります[5]。

I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	命令	LD^{PC}	S_1^A	S_0^A	S_1^B	S_0^B	S^C	S^O	$\overline{A}/S$
0	0	0	0	0	0	0	0	NOP	0	0	0	0	0	0	0	
0	0	0	0	0	0	1	0	OUT A	0	0	0	0	0	0	1	
0	0	0	0	0	1	0	1	CMP A, B	0	0	0	0	0	1	0	1
0	0	0	0	1	0	0	0	CPY B, A	0	0	0	0	1	0	0	
0	0	0	1	0	1	0	0	ADD B, A	0	0	0	1	0	1	0	0
0	0	0	1	0	1	0	1	XSUB B, A	0	0	0	1	0	1	0	1
0	0	0	1	1	0	0	0	IN B	0	0	0	1	1	0	0	
0	0	1	0	0	0	0	0	CPY A, B	0	0	1	0	0	0	0	
0	0	1	0	1	0	0	0	SWP A, B	0	0	1	0	1	0	0	
0	1	0	0	0	1	0	0	ADD A, B	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	1	SUB A, B	0	1	0	0	0	1	0	1
0	1	1	0	*data*				CPY A, *data*	0	1	1	0	0	0	0	
1	0	*address*						JPC *address*	*CF*	0	0	0	0	0	0	
1	1	*address*						JMP *address*	1	0	0	0	0	0	0	

[5]空欄の値は 0 でも 1 でも構いません。

このように、機械語命令のビットパターンを解読して、CPU の各部を制御する信号を生成する論理回路を一般に**デコーダ**と呼びます。この CPU のデコーダは次のような組み合せ論理回路で実現できます。



計算機システムⅠ・第13回・終わり