第３章　マイクロコンピュータ

1　目的

代表的な 8 ビット・マイクロ・コンピュータである Z80 を通して、マイクロ・コンピュータ・システムに関する基礎概念を修得することを目的とする。Z80 は 20 年以上前にインテル社の 8080 というマイクロコンピュータを改良したものとしてザイログ社で開発された。現在のいわゆるパーソナルコンピュータのきっかけとなったチップである。世界で沢山のメーカーがセカンドソースとして製造を行ってきており、現在も各種の装置へ組み込まれている。

身近な応用例としては「遊具の制御」や「 GBA 」などがある。現状では低価格であり、ASIC のライブラリーとしても利用され様々な機器の制御にいまだ使用されている。一般的にこの様な使用形態をソフトロジックなどと呼び、論理回路で固定的に論理動作を実現するのではなく、プログラミングで動作を目的に応じて容易に変更できる特徴を持っている。

この実習ではプログラムにより論理動作を実際に行うことの習得を行うが、プログラミングには細かな制御の記述が可能なアセンブリー言語を使用する。

2　マイクロコンピュータの概要

マイクロコンピュータは半導体の開発・製造技術の進歩によりその性能が急速に向上してきた。マイクロコンピュータは本来的なコンピュータとして WINDOWS などを動作させているが，その他にすでに述べた様に様々な装置の中に組み込まれている。この場合ではマイクロコンピュータは，何らかの計算を行うのではなく複雑な処理を小型な装置で行う。その様なマイクロコンピュータの利用例は豊富に存在している。

マイクロコンピュータの利用の一環に，ハードウエアロジックに対しソフトロジックと呼ばれるものがある。この手法によると複雑な処理ができるだけでなく、柔軟に要求に対応できる。また同一のハードウエア構成で異なった機能を，部品を交換したり半田付けを行う事なく実現することができる（これがソフトウエアーロジックという名称のいわれでもある。）。この様な目的でマイクロコンピュータを利用する場合、C に代表される様な「高級言語」を使用して処理を記述することはできる。しかし利用できるハードウエアの制約、（例えばメモリーの大きさ）さらにコストの制約などから一般にアセンブラーが使用される。

2.1 Z-80 マイクロコンピュータの構成

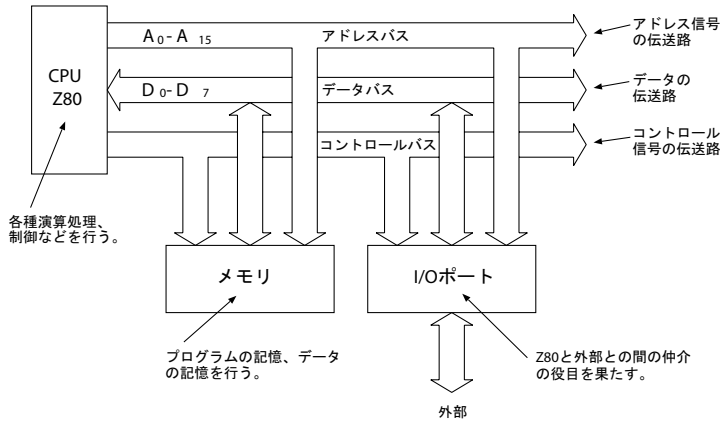


図 1: マイクロコンピュータの構成

マイクロコンピュータとして機能させるためには、一般に図 1 のような構成がとられる。そこでは 3 つの要素が必要とされる。

(1) CPU

コンピュータを構成するときに中心的な役割を果たすのが CPU である。ここでは Z80 といるマイクロコンピュータを使用する。CPU はメモリーに記録されているプログラムを順じ読み取り、その内容に従い Z80 は算術演算（加算、減算など)、論理演算（AND,OR,NOT, その他)、各種判断、データブロック転送やサーチなどを実行する。

(2) メモリ

プログラムやデータの記憶を行なう。メモリはコントロールバスからの信号に従って記憶内容をテータバスに読み出したり、データバスからデータを記憶する。また CPU のプログラム実行の時に一時的な記録を行ったり、サブルーチン実行のための戻りアドレスを記録する目的にも利用することができる。

(3) I/O ポート (入出力ポート)

マイクロコンピュータの内部と外部との境界の役割を果たす。Z80 で処理した結果を外部に出力したり、外部からのデータを内部に読み込むときのデータの出入口であると考えればよい。

(4) アドレスバス

Z80 がメモリの番地（データが記憶されている場所）を指定したり、いくつかある I/O ポートを指定する信号（アドレス信号）を送る伝送路である。アドレスバスは 16 本の線（16 ビット）の束で、各線には $A_0, A_1, \cdots, A_{15}$ という名前がついている。

(5) データバス

Z80、メモリ、I/O ポートの間でデータのやり取りを行なうときに用いられる伝送路である。データバスは 8 本（8 ビット）の線の束で、各線には $D_0, D_1, \cdots, D_7$ という名前がついている。

(6) コントロールバス

Z80 がメモリや I/O ポートなどを制御するため（例えばデータの書き込み）の指示を伝える伝送路である。

2.2 レジスタ

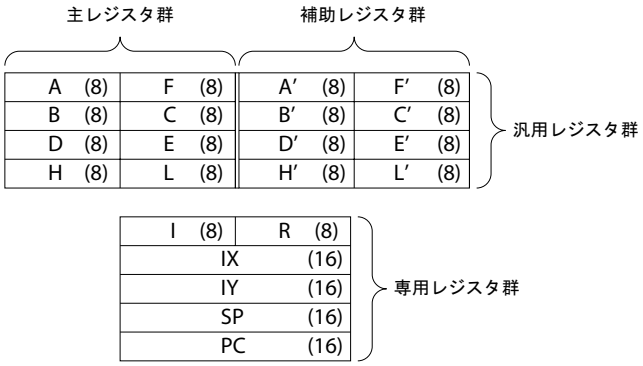


図 2: Z80 CPU の内部レジスタ群

レジスタは Z80 がプログラムを実行していく上で必要となる様々なデータの一時的な記憶場所として使用される。図 2 に Z80 の内部レジスタ群を示す。Z80 の内部レジスタ群は、主レジスタ群、補助レジスタ群、専用レジスタ群の 3 つから成っている。

2.2.1 主レジスタ群

主レジスタ群は 8 ビットのレジスタ 8 個から成っている。

A レジスタ

このレジスタは、通称アキュムレータと呼ばれる。データの処理に関しては、他のどれよりも機能が豊富で、処理も速くできるようになっている。演算のほとんどは、このアキュムレータを使って行えるし、データの読み込み、書き出しの中継点としても使うようにできている。

B レジスタ

データを一時保持しておく汎用レジスタのひとつであるが、特殊な機能もある。レジスタ B は、減数カウンタとして使え、この内容が 0 になると CPU の状態を変化させて、プログラムの流れを切替えることができる。

C レジスタ

汎用レジスタのひとつである。レジスタ C は、入出力のポート指定用に使うことができる。また、C レジスタは、B レジスタと組み合わせて、レジスタ・ペア BC としてアドレス指定に使ったり、減数カウンタとして使える。

D レジスタと E レジスタ

汎用レジスタである。D と E を各々別々に使用することもできるし、B,C と同様、レジスタ・ペア DE としてメモリのアドレス指定としても使用できる。

H レジスタと L レジスタ

汎用レジスタである。H と L を各々別々に使用することもできるし、B,C や D,E と同様 16 ビットのレジスタ・ペア HL として使える。しかし、メモリのアドレス指定に対しては、BC や DE よりも機能が豊富になっている。

F レジスタ

F レジスタの内容は、主として各種演算が行われるごとに演算結果に応じて変化する。F レジスタは 8 ビットだが、実際には図 3 に示すように 6 つの場所が利用される。演算結果に応じて、これら 6 つの場所の値が自動的に影響を受ける。これら 6 つの場所 S,Z,H,P/V,C の各々を総称して**フラグ**という。これらのフラグは影響を受けるまで値を保持し続ける。

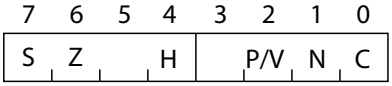


図 3: F レジスタ

・S：(サインフラグ)
結果が正であったか負であったかを記憶する。演算の結果最上位ビット（サインビット）が 1 ならば 1、0 ならば 0 になる。

・Z：(ゼロフラグ)
結果がゼロであったかゼロでなかったかを記憶する。演算結果が 0 になったとき 1、そうでなければ 0 となる。

・H：(ハーフキャリー)
演算の結果、ビット 3 からビット 4 へのキャリーが生じたか否かを記憶する。ただし、このフラグは DAA という命令を実行するときに CPU が自動的に使用するフラグで、プログラムによって判断材料として使用することはできない。

・P/V：(パリティー/オーバーフロー)
演算結果の中にある 1 の個数が偶数であったか奇数であったかを記憶する。また、演算結果がオーバーフローしたか否かにも利用される。P フラグとして使用されるか V フラグとして使用されるかは、プログラム中の命令による。主として論理演算のときはパリティ P（結果の 1 の個数が偶数なら P=1、奇数なら P=0）として用いられる。算術演算のときはオーバーフローフラグ V として用いられる。オーバーフローフラグは最上位（符号）ビットに関して定まる。以下に示す関係式の ( ) 内は符号ビットとする。

$$\text{加算：}(0)+(0)=(1)\ \text{または}\ (1)+(1)=(0)\ \text{のとき V=1}$$

$$\text{減算：}(0)-(1)=(1)\ \text{または}\ (1)-(0)=(0)\ \text{のとき V=1}$$

これ以外のとき V=0（オーバーフローでない）となる。

・N：(加/減算フラグ)
演算内容が加算であったか減算であったかを記憶する。加算命令を実行したとき N=0、減算命令を実行したとき N=1 となる。N フラグは H フラグと同様 DAA という命令を Z80 が実行するときに自動的に利用される。プログラムを進める上での判断材料としては利用されないフラグである。

・C：(キャリーフラグ)
演算した結果キャリーが生じたか否かを記憶する。C を CY ともかく。キャリーフラグが影響を受けるのは、加算、減算、シフト、ローテイト、補正のいずれかの命令を実行したときである。

加算（ADD,ADC）の場合は、最上位ビットから桁上がり（キャリー）が生じたとき C=1、そうでなければ C=0 となる。

減算（SUB,SBC）の場合は、最上位ビットを越えて桁借り（ボロー）が生じたとき C=1、そうでなければ C=0 となる。

シフト／ローテイト（RLC,RL,RRC,RR,SLA,SRA,SRL）の場合は、レジスタあるいはメモリからシフトされてきた値がそのまま C のなかに書き込まれる。

補正（DAA,NEG,CCF,SCF) の場合、DAA 実行前の C,H フラグおよび $A_7 \sim A_4$,$A_3 \sim A_0$ の内容により DAA 実行後の C フラグは 1 か 0 になる。NEG は実行前のアキュムレータの内容が 0 以外のとき C=1、0 のとき C=0 となる。CCF によって C の内容は反転、SCF によって C=1 となる。

以上述べたフラグのうち、S,Z,P/V,C の 4 つのフラグが条件ジャンプ命令、条件コール命令、条件リターン命令などで直接利用されるものである。この 4 つフラグを利用することで、CPU は様々な判断能力を発揮することになる。

2.2.2 補助レジスタ群

補助レジスタ群は主レジスタ群と同じレジスタの構成になっている。表現上でこれらを区別するため、補助レジスタ群の方にはダッシュ記号がついている。

補助レジスタの内容を直接操作する命令はないが、主レジスタ群の内容と補助レジスタ群の内容を交換する命令があるので、それらの命令を使っていったん主レジスタ群に補助レジスタ群の内容を移せば、主レジスタ用の命令でデータ処理ができる。主レジスタ群と補助レジスタ群の間でデータ交換を行う (正確にはレジスタ自体の交換を行っている) 命令をあげておく。

- **EX AF,AF'** (A と A'、F と F' の内容の交換)
- **EXX** (B と B'、C と C'、D と D'、E と E'、H と H'、L と L' の内容の交換)

これらのレジスタ交換はフラグの内容に影響を与えることなく実行される。特に、サブルーチンコールで 1 回だけレジスタの内容を退避させればよい場合に使うと便利である。

2.2.3 専用レジスタ群

専用レジスタ群は 2 つの 8 ビットのレジスタと 4 つの 16 ビットのレジスタから、成っている。これらの中で SP（スタックポインター）と PC（プログラムカウンター）はプログラムの実行に対し重要な役割を持っている。

I レジスタ

Z80 の割り込みモード 2 のときに使用される。詳しい説明は省く。

R レジスタ

Z80 外部にダイナミック RAM を接続したときのリフレッシュ用に用いられる。詳しい説明は省く。

IX,IY レジスタ

IX,IY は、インデックス・レジスタと呼ばれる。レジスタペア BC、DE、HL と同様にアドレス指定を行うために使用する。

Z80 のインデックス・レジスタ IX、IY は各々独立に使える上、ディスプレイスメント (偏位) が付加できる。

インデックス・レジスタの内容に、ディスプレイメントの値を加えた値で、メモリの番地を指定できる。

例えば次のような命令を図にすると、図 4 のようになる。

LD A,(IX+5) ； インデックスレジスタ IX の内容に 5 を加えた値で指定されるメモリロケーションからデータをアキュームレータにロードせよ。

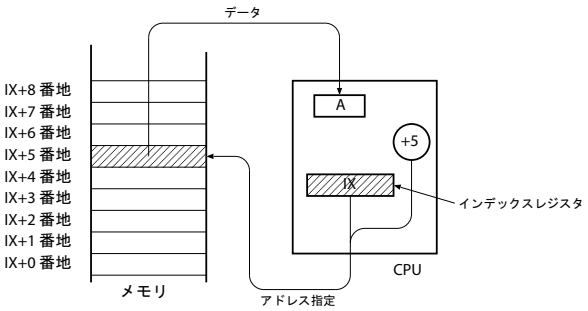


図 4: インデックスレジスタによるアドレシング

このレジスタは、たとえばデータのテーブル (表) を作る場合や、そのテーブルからデータを読み出す場合に便利に使える。レジスタ IX,IY は、上のような使い方をしない時は、単にデータ保存用のレジスタとしても使うことができる。

SP （スタック・ポインタ）

SP(スタック・ポインタ) は特殊なアドレス用レジスタで、メモリのアドレスを意識せずにメモリから、あるいはメモリへデータを出し入れするために使用される。その典型的な例としてサブルーチン実行時の戻りアドレスの記録に使用される。書き込みや読み出しを行うとメモリーのアドレスを示すポインターは自動的に増減する。

このポインタで指定するメモリをスタックと呼ぶが、次のような便利な機能が備わっている。

例えば、次のような命令でレジスタ・ペア BC の内容をスタックへ入れることができる。その結果、スタック・ポインタ SP の内容は、自動的に 2 だけ減少する。

PUSH BC ； レジスタ B と C の内容を、スタックポインタ SP で指定されるメモリロケーションに入れ、スタックポインタの値を 2 だけ減らせ。

次の命令は、その逆の操作ができる。

POP BC ； ポインタ SP の内容で指定されるメモリ・ロケーションから、レジスタ B と C にデータをロードし、ポインタの値を 2 だけ増せ。

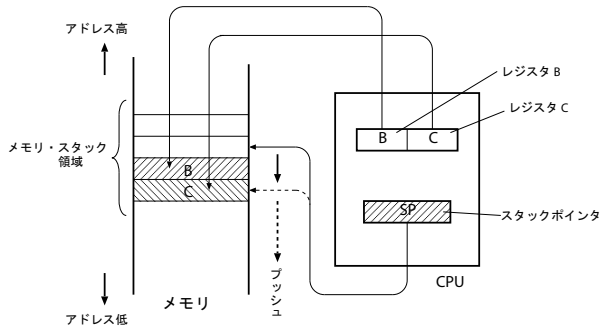


図 5: スタックポインタの働き

このスタックポインタを使えば、データを順に（アドレスを気にせずに）押し込んでいったり、引張り出してきたりすることができるので大変便利である。

だだし、プログラムを組む場合、必ず前もって RAM の内でスタック領域を決め、その一番高いアドレスの値をスタック・ポインタに入れておく必要がある。

PC （プログラム・カウンタ）

PC（プログラム・カウンタ）は、メモリのプログラム領域から命令を読み出してくる際に、そのメモリアドレスを指定するためのものである。

このカウンタは、次のような性質を持っている。

- CPU にリセットをかけると、プログラム・カウンタの内容は、0(ゼロ) となる。プログラム・カウンタの内容が 0 ということは、メモリの 0000H 番地を指すことを意味する。
- このカウンタの内容は、機械語（1 バイト分）を読む段階で自動的に 1 ずつ増されていく。例えば 0000H からスタートすれば、0001H → 0002H → 0003H 番地と繰り上げられていく。
- プログラム・カウンタの内容は一般に分岐の機能を持った命令で書き換えられる。

例） **JP ABC** ； プログラム・カウンタ PC に、ABC という所の番地を入れてそこから続けよ。

プログラム・カウンタに新しい内容が入ると、プログラム中の命令はその新しい内容で指定される場所（メモリ・ロケーション）から、順次読みだされ実行されていく。

例） **CALL ABC** ； プログラム・カウンタ PC の内容をスタック・ポインタで指定されるメモリ・ロケーションに保存し、プログラム・カウンタに、ABC という番地を入れプログラムの流れをジャンプする。

実際にプログラムを作成していくときには、特にプログラム・カウンタそのものを意識する必要はない。

2.3 命令と機械語

ここでは命令（ニモニック）から機械語を求める手順を例で示しておく。

ニモニックとして、**LD r,r'** が与えられたとする。この命令ではレジスター r' からレジスター r にデータがコピーされるが、これを機械語に直したバイナリーは図 6 に示す構造になっている。上位 2 ビットが 01 で残り 6 ビットのうち 3 ビットごとに **r,r'** となっている。**r,r'** のところには Z80 が持っている汎用レジスタ名を書き込む。命令でいうと例えば

LD B,A

である。この **B** が **r**、**A** が **r'** に相当する。これを機械語に直すには **B,A** のコードを求めればよい。**B** は 000、**A** は 111 と約束されている。従って、求める機械語は図 6 に示すように 01000111 となる。16 進で表せば 47H(H は 47 が 16 進であることを意味する補助的な記号）となる。

この例のように機械語が 1 バイトのものを 1 バイト命令という。この他に 2 バイト命令、3 バイト命令、4 バイト命令がある。これらはすべて機械語が何バイトかを意味している。

また命令表のなかに T ステートという表現があるが、これは命令を実行するときに必要なクロックパルスの個数を意味する。したがって、クロックパルスの周期がわかれば各命令の実行時間を算出できる。

2.4 8 ビットロードの命令セット

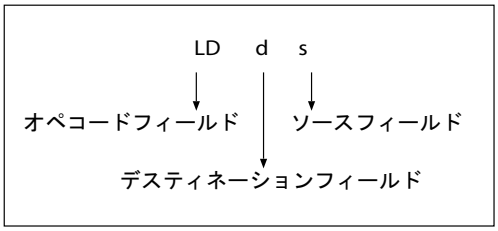
Z80 内部のレジスタ群、メモリの間で 8 ビット単位でデータのやりとりを指示する。

8 ビットロード命令は

レジスタのコード

r, r'	レジスタ
000	B
001	C
010	D
011	E
100	H
101	L
111	A

図 6: 命令と機械語



という形になっている。

ソースフィールドはレジスタ、メモリの番地、あるいは 8 ビットのデータを意味する。

デスティネーションフィールドはレジスタ、またはメモリの番地を示す。

LD はソースフィールドで指定されたレジスタの内容、メモリの内容または 8 ビットのデータをデスティネーションフィールドに指定されているレジスタまたはメモリに書き込む意味を持っている。ソース側のレジスタやメモリの内容はこの命令を実行しても変わらない。

なお、この命令は 8 ビットデータのやりとりを行うので、Z80 が持つレジスタ群として 8 ビット単位で使用できるものが対象となる。

LD r, r' の例

命令　　LD A,C

機械語　　01111001 (2 進)　　79 (16 進)

(1 バイト)

働き　　C レジスタの内容が A レジスタに書き込まれる。
ただし、C レジスタの内容はもとのままである。

LD r, n の例

命令　　LD B, 5DH

機械語　　00000110　　06

(2 バイト)　　01011101 (2 進)　　5D (16 進)

働き　　命令の 2 バイト目 5D が B レジスタに書き込まれる。
したがって B レジスタの内容は 5D となる。

LD r, (HL) の例

命令　　LD A, (HL)

機械語　　01111110 (2 進)　　7E (16 進)

(1 バイト)

働き　　仮に HL の内容が 1F04H であれば、メモリの 1F04H 番地の
内容が A に書き込まれる。

LD (HL), r の例

命令　　LD (HL), A

機械語　　01110111 (2 進)　　77 (16 進)

(1 バイト)

働き　　仮に HL の内容が 1F04H であれば、メモリの 1F04H 番地に
A の内容が書き込まれる。

LD (HL), n の例

命令　　LD (HL), 51H

機械語　　00110110　　36

(2 バイト)　　01010001 (2 進)　　51 (16 進)

働き　　仮に HL の内容が 1F04H であれば、メモリの 1F04H 番地に
命令の 2 バイト目 51H が書き込まれる。

LD A, (nn) の例

命令	LD A, (20C3H)	
機械語	00111010	3A
(3 バイト)	11000011	C3
	00100000 (2 進)	20 (16 進)
働き	メモリの 20C3H 番地の内容が、 A レジスタに書き込まれる。	

LD (nn), A の例

命令	LD (20C3H), A	
機械語	00110010	32
(3 バイト)	11000011	C3
	00100000 (2 進)	20 (16 進)
働き	メモリの 20C3H 番地に A レジスタの 内容が書き込まれる。	

2.5 16 ビットロードの命令セット

Z80 内部の 16 ビット幅のレジスタ、メモリ間での 2 バイトを単位としたデータのやりとりを指示する。

命令の読み方と機械語への変換の例を示しておこう。

LD dd, nn

これは 2 バイトのデータ nn を 16 ビットのレジスタ dd に書き込めという意味である。dd は命令表中にあるようにレジスタペア BC,DE,HL またはスタックポインタ SP のことである。また nn は 2 バイトすなわち 4 桁の 16 進数で表された値である。

LD HL, 2057H

この命令の構造は

00dd0001
← n → (下位 8 ビットを書く)
← n → (上位 8 ビットを書く)

である。

HL は命令表より 10 とすることがわかる。従って機械語は

00100001
01010111
00100000

となる。

この命令を実行すると、H レジスタに 20H が書き込まれ、L レジスタに 57H が書き込まれる。

LD HL, (nn) の例

命令	LD HL, (1234H)	
機械語	00101010	2A
3 バイト	00110100	34
	00010010 (2 進)	12 (16 進)
働き	メモリの 1235H 番地の内容が H レジスタに、1234H 番地の内容が L レジスタに書き込まれる。	

LD (nn), HL の例

命令	LD (1234H), HL	
機械語	00100010	22
3 バイト	00110100	34
	00010010 (2 進)	12 (16 進)
働き	メモリの 1235H 番地に H レジスタの内容が、1234H 番地に L レジスタの内容が書き込まれる。	

LD SP, HL

機械語	11111001 (2 進)	F9 (16 進)
1 バイト		
働き	H レジスタの内容が SP の上位 8 ビットに書き込まれ、L レジスタの内容が SP の下位 8 ビットに書き込まれる。	

PUSH qq の例

命令	PUSH AF	
機械語	11110101 (2 進)	F5 (16 進)
1 バイト		
働き	仮に SP の内容が 7F05H であったとすれば、 **1.** 7F04H 番地に A の内容が書き込まれ、 **2.** 7F03H 番地に F の内容が書き込まれる。 **3.** SP は 7F03H になる。	

POP qq の例

命令	POP DE	
機械語	11010001 (2 進)	D1 (16 進)

1 バイト

働き　　仮に SP の内容が 7F03H であったとすれば、

1. 7F03H 番地の内容が E に書き込まれ、

2. 7F04H 番地の内容が D に書き込まれる。

3. SP は 7F05H になる。

2.6　8 ビットの算術論理演算の命令セット

8 ビットの算術演算命令として、

ADD, ADC, SUB, SBC

の様な命令がある。

また 8 ビットの論理演算命令として、

AND, OR, XOR

がある。そしてこれらの命令の書式は ADD を例にすると次のようになっている。

ADD d, s

ここで s（ソース）として、レジスタ、メモリ、データが指定できる。d（デスティネーション）にはアキュムレータのみが指定できる。

以上の命令の他に、CP（compare）という比較命令がある。この命令はフラグの内容のみに影響を与える命令である。

また、+1 か −1 を行なう場合には、INC,DEC 命令を用いればよいであろう。これらの命令はレジスタ、メモリに対して用いることができる。

最後に注意することは、P/V フラグである。算術演算の場合 P/V フラグは V フラグすなわちオーバーフローしたか否かを示すフラグとなる。一方論理演算に対しては P/V フラグは P（パリティ）すなわち結果の中に含まれる 1 の個数が偶数か奇数かを示すフラグとなる。

ADD A, r の例

命令　　ADD A, L

機械語　　10000101 (2 進)　　85 (16 進)

1 バイト

働き　　A の内容が 02H、L の内容が 15H であったとする。

1. A の内容と L の内容の和 17H が、

2. A に書き込まれる。

したがって、A の内容は 02H から 17H に変わる。

3. フラグは

C=0, Z=0, P/V=0, S=0, N=0, H=0

ADD A, n の例

命令　　ADD A, 13H
機械語　　11000110　　C6
1 バイト　　00010011 (2 進)　　13 (16 進)
働き　　A の内容が 70H であったとする。
1. A の内容 70H と 13H との和 83H が、
2. A に書き込まれる。
したがって、A の内容は 70H から 83H に変わる。
3. フラグは
C=0, Z=0, P/V=1, S=1, N=0, H=0

ADD A, (HL)

機械語　　11000110 (2 進)　　86 (16 進)
1 バイト
働き　　A の内容が 74H、HL の内容が 9000H、
9000H 番地の内容が 8CH であったとする。
1. A の内容 74H と (HL) 番地の内容 8CH の和 00H が、
2. A に書き込まれる。
したがって、A の内容は 74H から 00H に変わる。
3. フラグは
C=1, Z=1, P/V=0, S=0, N=0, H=1

ADC A, s の例

命令　　ADC A, D
機械語　　10001010 (2 進)　　8A (16 進)
1 バイト
働き　　A の内容が 70H、D の内容が 0FH、
CY（キャリーフラグ C）が 1 であったとする。
1. A の内容と D の内容と CY の和 80H が、
2. A に書き込まれる。
したがって、A の内容は 70H から 80H に変わる。
3. フラグは
C=0, Z=0, P/V=1, S=1, N=0, H=1

SUB s の例

命令	SUB (IX+05H)	
機械語	11011101	DD
3 バイト	10010110	96
	00000101 (2 進)	05 (16 進)

働き　A の内容が 60H、IX の内容が 9000H、
9005H 番地の内容が D0H であったとする。
1. IX と d（05H）の和によってメモリの 9005H 番地
が指定され、
2. A の内容 60H と 9005H 番地の内容 D0H の差 90H が、
3. A に書き込まれる。したがって、A の内容は 90H となる。
4. フラグは
C=1, Z=0, P/V=1, S=1, N=1, H=0

SBC n の例

命令	SBC A, 90H	
機械語	11011110	DE
2 バイト	10010000 (2 進)	90 (16 進)

働き　A の内容が A0H、CY の内容が 1 であったとする。
1. A−n−CY の値すなわち、A0H−90H−1 の値 0FH が
2. A に書き込まれる。したがって、A の内容は A0H から
0FH に変わる。
3. フラグは
C=0, Z=0, P/V=0, S=0, N=1, H=1

AND n の例

命令	AND F0H	
機械語	11100110	E6
2 バイト	11110000 (2 進)	F0 (16 進)

働き　A の内容が 9CH であったとする。
1. A の内容 9CH と F0H の AND の結果 90H が
2. A に書き込まれる。したがって、A の内容は 9CH から
90H に変わる。
3. フラグは
C=0, Z=0, P/V=1, S=1, N=0, H=1

1 0 0 1 1 1 0 0 (9CH)
AND) 1 1 1 1 0 0 0 0 (F0H)
1 0 0 1 0 0 0 0 (AND の結果)

このように、AND 演算は、少なくとも一方が 0 のところの結果は 0 となり、両者が 1 のところだけが 1 となる。

OR n の例

命令 OR F0H
機械語 11110110 F6
2 バイト 11110000 (2 進) F0 (16 進)
働き A の内容が 9CH であったとする。
1. A の内容 9CH と F0H の OR の結果 FCH が
2. A に書き込まれる。したがって、A の内容は 9CH から FCH に変わる。
3. フラグは
C=0, Z=0, P/V=1, S=1, N=0, H=0

1 0 0 1 1 1 0 0 (9CH)
OR) 1 1 1 1 0 0 0 0 (F0H)
1 1 1 1 1 1 0 0 (OR の結果)

このように、OR 演算は、少なくとも一方が 1 のところの結果は 1 となり、両者が 0 のところだけが 0 となる。

XOR n の例

命令 XOR F0H
機械語 11101110 EE
2 バイト 11110000 (2 進) F0 (16 進)
働き A の内容が 9CH であったとする。
1. A の内容 9CH と F0H の XOR の結果 6CH が
2. A に書き込まれる。したがって、A の内容は 9CH から 6CH に変わる。
3. フラグは
C=0, Z=0, P/V=1, S=0, N=0, H=0

1 0 0 1 1 1 0 0 (9CH)
XOR) 1 1 1 1 0 0 0 0 (F0H)
0 1 1 0 1 1 0 0 (XOR の結果)

このように、XOR 演算は、両者が共に 1 または共に 0 ところは 0

となり、両者が互いに異なる値になっているところは 1 となる。

CP n の例

命令	CP F0H	
機械語	11111110	FE
2 バイト	11110000 (2 進)	F0 (16 進)

働き　A の内容が 9CH であったとする。
1. A の内容 9CH と F0H の比較（A と F0H の差）結果が
2. フラグとして残される。A の内容は変化しない。
フラグは、
C=1, Z=0, P/V=0, S=1, N=1, H=0

INC s の例

命令	INC B	
機械語	00000100 (2 進)	04 (16 進)
1 バイト		

働き　B の内容が 26H であったとすれば、
1. B の内容は 1 だけ増加して 27H となる。
2. フラグは、
C=変化しない, Z=0, P/V=0, S=0, N=0, H=0

DEC s の例

命令	DEC (HL)	
機械語	00110101 (2 進)	35 (16 進)
1 バイト		

働き　HL の内容が 16F3H で、16F3H 番地の内容が 26H であったとすれば、
1. (HL) 番地すなわち 16F3H 番地が指定され、
2. 16F3H 番地の内容 26H が 1 だけ減じられて 25H となり、
3. この 25H が 16F3H 番地に書き込まれる。したがって、
16F3H 番地の内容は 26H から 25H に変化する。
4. フラグは、
C=変化しない, Z=0, P/V=0, S=0, N=1, H=0

2.7　16 ビット算術演算の命令セット

8 ビットの算術演算はアキュムレータがデスティネーションとなっていたが、16 ビットの算術演算では HL レジスタ、IX レジスタ、IY レジスタがデスティネーションとなっている。特に ADD と ADC を用いることで、32 ビット以上の符号つき 2 進数の加算が容易に実現できる。

また、BC,DE,HL,SP,IX,IY に対し +1 および −1 を行う命令 INC,DEC がある。ただし、これらのレジスターに対して INC,DEC 命令を用いてもフラグには何の影響もない。

2.8　ジャンプの命令セット

Z80 のジャンプ命令には、絶対アドレッシング（命令コード中のアドレスフィールド nn で指定したアドレスそのものにジャンプする）、相対アドレッシング（現在のプログラムカウンタの値に対し、どれだけジャンプするかのアドレス値の差を与えてジャンプする）、間接アドレシング（HL,IX,IY の内容をアドレスとみなし、そのアドレスにジャンプする）の 3 通りのタイプに大別される。

命令セット中、nn は上で述べたようにアドレスを表す。cc は条件付きジャンプ命令用の条件である。このときの条件としてフラグ S,Z,P/V,CY が用いられる。e は相対ジャンプ幅である。

また、DJNZ e は相対ジャンプ命令であるが、B レジスタに対して 1 だけ減じた後、結果が 0 であれば次へ、0 でなければ e で指定した幅だけジャンプする。

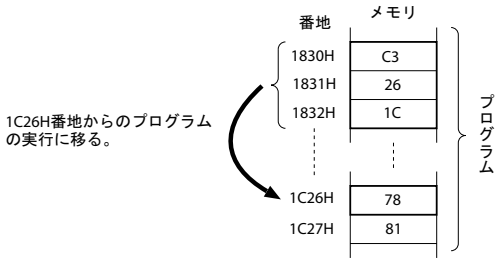
DJNZ e が有効に利用できる処理手順はよくプログラムの中で見かけることができる。この命令は多くの繰り返し処理に使用されている。例えばあるメモリーの領域から別の領域にデータのコピーを行う場合などに使用される。

JP nn の例

命令	JP 1C26H	
機械語	11000011	C3
3 バイト	00100110	26
	00011100 (2 進)	1C (16 進)

働き

無条件に指定されたアドレスに制御を移す。具体的には図で示す様な動作をする。

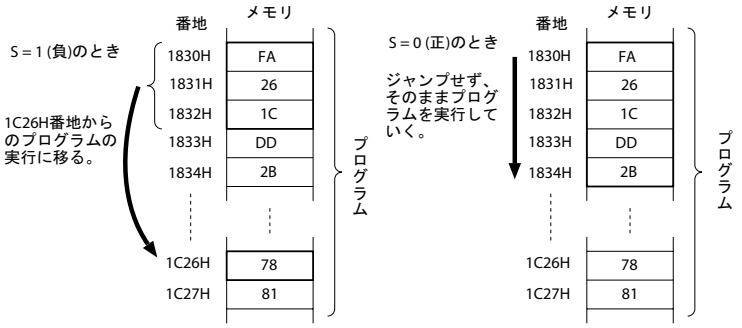


JP cc, nn の例

命令	JP M, 1C26H	
機械語	11111010	FA
3 バイト	00100110	26
	00011100 (2 進)	1C (16 進)

働き

cc で示されるフラグの状態に従いプログラムの制御を移す。
例では M フラグが立っている場合はプログラムは指定されたアドレスに分岐するが、
それ以外の場合はこの命令の後の命令を実行する。

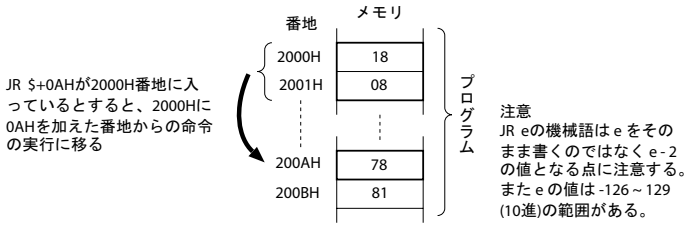


JR e の例

命令	JR $+0AH	
機械語	00011000	18
2 バイト	00001000 (2 進)	08 (16 進)

働き

この命令は、動作を指定された相対的なアドレスにプログラムを無条件に分岐する。



$+0AH は現在の位置から 10 バイト分先へジャンプする意味である。
仮に$−0AH なら 10 バイト分後へもどる意味になる。

JR C, e の例

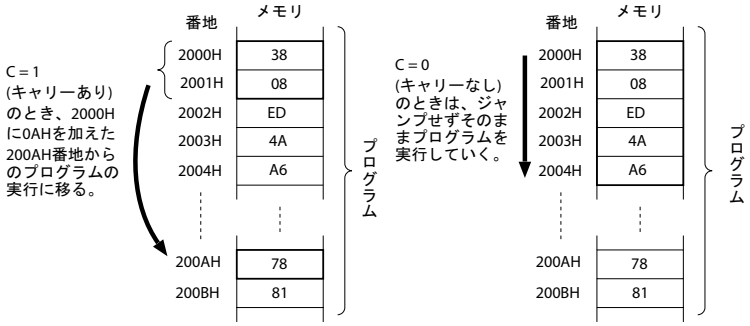
命令　　JR C, $+0AH

機械語　　00111000　　38

2 バイト　　00001000 (2 進)　　08 (16 進)

働き

この命令は、動作を指定された相対的なアドレスにプログラムを指定された条件に従い分岐する。



JP (HL)

機械語　　11101001 (2 進)　　38 (16 進)

1 バイト

働き　　JP (HL) を実行する時点で、HL の内容が 50F3H であれば、
50F3H 番地からのプログラムの実行に移る。

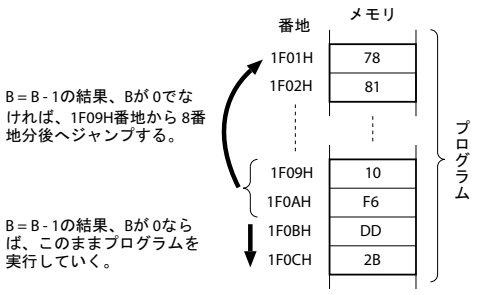
DJNZ e の例

命令　　DJNZ $−8

機械語	00010000	10
2 バイト	11110110 (2 進)	F6 (16 進)

働き

B レジスターの内容を -1 し、結果が 0 でなければ e で指定された大きさだけ分岐する。



$−8 の機械語部分の求め方

1. e−2 を求める。
この場合 e=−8 したがって e−2 は −10 となる。

2. 10 の 2 進数を求める。
0 0 0 0 1 0 1 0

3. −10 に相当する 2 進数 (補数) を求める。

$$\begin{array}{rr} & 1\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ -) & 0\,0\,0\,0\,1\,0\,1\,0 \\ \hline & 1\,1\,1\,1\,0\,1\,1\,0 \end{array}$$

この 1 1 1 1 0 1 1 0 が求める$−8 の機械語である。
16 進ならば F6H となる。

2.9 サブルーチンコールとリターンの命令セット

サブルーチンコールとしては、

CALL nn

CALL cc, nn

の 2 種類がある。上の CALL はいわば無条件コール命令で、下は条件付コール命令である。

サブルーチンからのリターンとしては、

RET

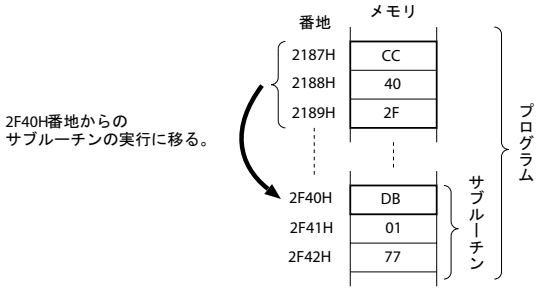
RET cc

などがあり、RET は無条件リターンを、RET cc は条件付リターンを行う。

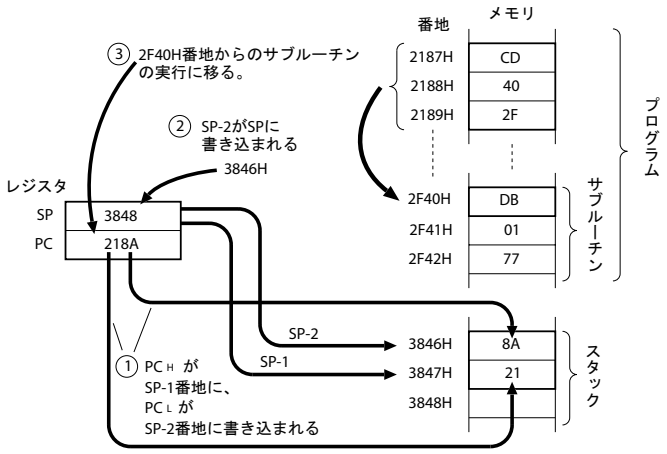
CALL nn の例

命令	CALL 2F40H	
機械語	11001101	CD
3 バイト	01000000	40
	00101111 (2 進)	2F (16 進)

働き （概要）



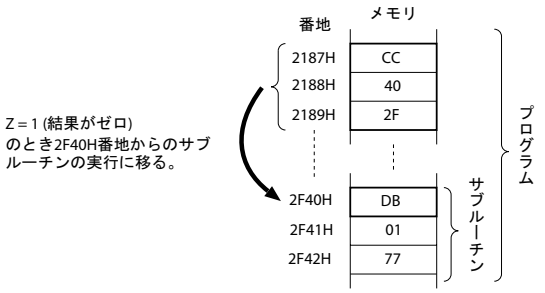
働き （詳細）

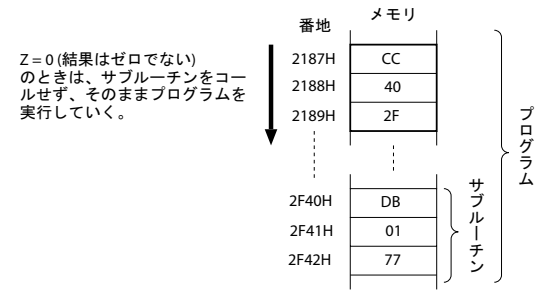


CALL cc, nn の例

命令 CALL Z, 2F40H

機械語	11001100	CC
3 バイト	01000000	40
	00101111 (2 進)	2F (16 進)
働き	条件が成立したときにサブルーチンをコールする。	
	サブルーチンのコールの手順は CALL nn と同じである。	



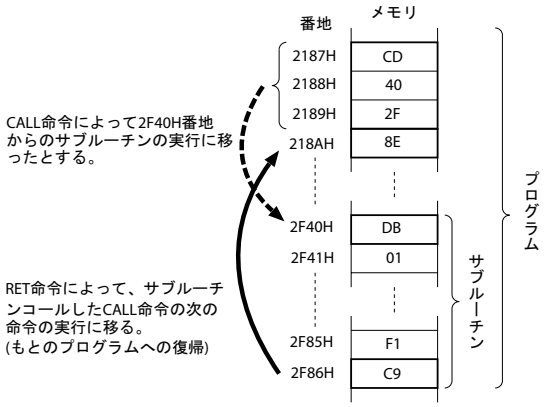


RET

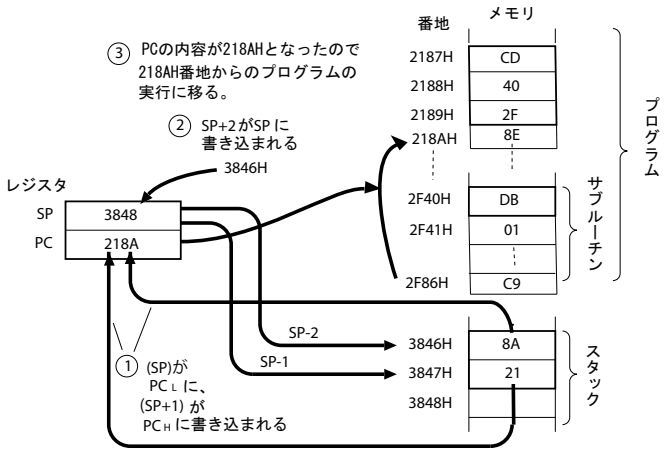
機械語　　11001001 (2 進)　　C9 (16 進)

1 バイト

働き （概要）



働き （詳細）



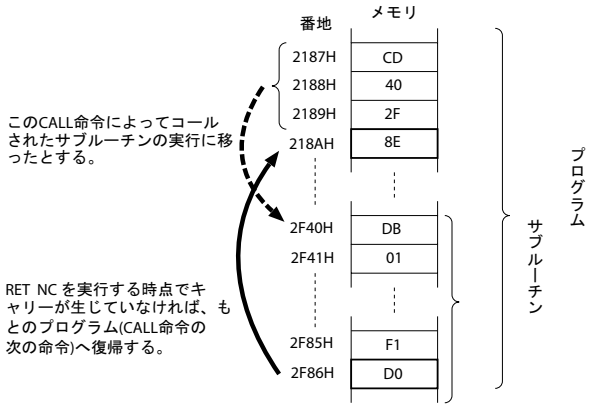
RET cc の例

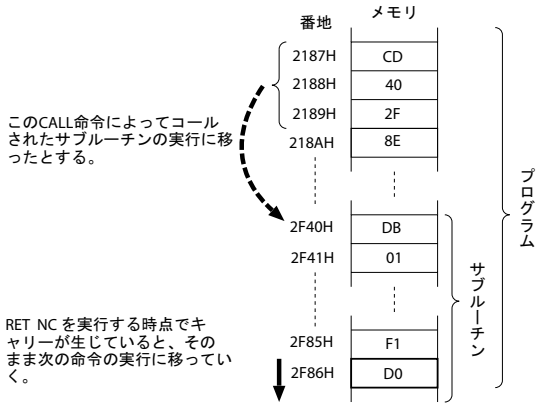
命令　　RET NC

機械語　　11010000 (2 進)　　D0 (16 進)

1 バイト

働き　　条件が成立したときに、サブルーチンからもとのプログラムへ復帰する。復帰の手順は RET と同じである。





2.10 入力出力の命令セット

IN A, (n) の例

命令	IN A, (80H)	
機械語	11011011	DB
2 バイト	10000000 (2 進)	80 (16 進)
働き	80H 番の入力ポートからのデータをアキュムレータに読み込む。	

IN r, (C) の例

命令	IN D, (C)	
機械語	11101101	ED
2 バイト	01010000 (2 進)	50 (16 進)
働き	レジスタ C の内容を入力ポートアドレスとし、その入力ポートからのデータを r（この例では D レジスタ）に読み込む。	

OUT (n), A の例

命令	OUT (20H), A	
機械語	11010011	D3
2 バイト	00100000 (2 進)	20 (16 進)
働き	IN A, (n) の命令と逆の働きをする。A（アキュムレータ）の内容を 20H 番の出力ポートに書き出す。	

OUT (C), r の例

命令	OUT (C), E	
機械語	11101101	ED
2 バイト	01011001 (2 進)	59 (16 進)
働き	IN r, (C) と逆の働きをする。r（この場合は E レジスタ）の内容を C レジスターで示されたアドレス（仮に C レジスタの内容が 10H であったとすれば、10H 番）の出力ポートに書き出す。	

3　実験課題

課題 1

課題では予め用意された機械語のプログラムを入力し，実際の動作を確かめなさい。なおアドレス 8102 と 8103 の内容を変更し、どの様に動作が変化するか観測しなさい。

このプログラムを実行すると，演習用のコンピュータについている LED に表示される数字が約 0.7 秒ごとに数が変化する。以下に記載したプログラムのリストの一番左は，２バイトのメモリーアドレスであり次の２桁の数字は１６進数で表された機械語である。命令の大きさは１バイトから３バイトの長さとなっている。リストの右側は、コンピュータへの命令を人間が理解しやすい形で表しており一般にアセンブラー言語と呼ばれている。この記述には機械語に変換されない命令も含まれており、これを擬似命令と呼んでいる。この擬似命令はアセンブリー言語から機械語へ変換する時の補助として使用される。以下にこのプログラムで使用されている擬似命令を説明しておく。

```
DISPLAY  EQU 0040H  ->  DISPLAY という文字列を１６進数の４０として扱う。

ORG      8000H      ->  １６進数の８０００番地からはじめる事を示す。
START:              ->  START という名称のラベル
END                 ->  アセンブラーの記述の終了を意味します。
```

ここではどの様な原理で数字が約 0.7 秒ごとに変化するかを理解しておく必要がある。

```
番地  機械語     Z80 アセンブリー言語
          ORG 8000H
          DISPLAY   EQU  0040H

          START:
8000  0E 00     LD C,0
          L1:
8002  79        LD A,C
8003  32 F5 FF  LD (0FFF5H),A  ; 表示用バッファーにデータ転送
8006  CD 40 00  CALL DISPLAY   ; 表示
8009  CD 00 81  CALL WAIT      ; 約 0.7 秒待ち
800C  0C        INC C          ;
```

```
800D  C3 02 80  JP L1           ; 繰り返し

;
; 時間待ちサブルーチン
;
        ORG 8100H
        WAIT:
8100  C5        PUSH BC
8101  01 00 00  LD BC,0
        WAIT1:
8104  0B        DEC BC
8105  78        LD A,B
8106  B1        OR C
8107  C2 04 81  JP NZ,WAIT1
810A  C1        POP BC
810B  C9        RET
        END
```

課題２

課題１のプログラムを実行すると、使用しているコンピュータの LED ディスプレーは約 0.7 秒で表示される数字がインクルメントされる。この時間を 2.1 秒になる様に変更しなさい。課題１では，サブルーチン WAIT により 0.7 秒の待ち時間を作っているが、これを応用して 2.1 秒の待ち時間となるプログラムを作成する必要がある。

[実施についての注意事項]

Z80 で使用されるアセンブリー言語のプログラミングでは以下の様な約束がある。

1) LD A,00H は，右がわに指定されたレジスターに１６進数の 00 を入れることを意味する。

2) LD (8400H),A は ( ) で示されたメモリーのアドレスに A レジスターの内容を転送することを意味する。

3) LD B,(HL) は HL レジスター（１６ビット幅であることに注意）の内容が示すメモリーのアドレスの内容を，B レジスターに転送する。

4) LD (HL),A は HL レジスターに内容が示すメモリーのアドレスに A レジスターの内容を転送する。

課題３

以下に示すアセンブリー・プログラムを 8000H 番地以降に作成し、ステップ・モードを用いて各レジスターの内容、メモリの内容などを確認しながら実行しなさい。

(1)

```
ORG 8000H
LD A,00H
LD (8400H),A
LD A,85H
LD (8401H),A
LD A,2Ah
LD (8500H),A
LD HL,(8400H)
LD B,(HL)
LD C,B
HALT
END
```

上記のプログラムを機械語に変換し、Z 80 のステップモードを用いて各レジスター、メモリの内容を確認しなさい。

プログラム	機械語	各レジスターの内容					各メモリーの内容		
		A	H	L	B	C	8400H	8401H	8500H
START:LD A,00H									
LD (8400H),A									
LD A,85H									
LD (8401H),A									
LD A,2AH									
LD (8500H),A									
LD HL,(8400H)									
LD B,(HL)									
LD C,B									
RET									

(2)

```
ORG 8000H
LD A,82H
INC A
LD B,A
```

```
        INC A
        LD H,A
        LD L,00H
        LD A,37H
        LD (HL),A
        HALT
        END
```

上記のプログラムを機械語に変換し、Z 80 のステップモードを用いて各レジスター、メモリの内容を確認しなさい。

プログラム	機械語	各レジスターの内容					メモリーの内容
		A	H	L	B	C	8400H
START:LD A,82H							
INC A							
LD B,A							
INC A							
LD H,A							
LD L,00H							
LD A,37H							
LD (HL),A							
HALT							

課題４

以下に示すアセンブラ・プログラムを 8000H 番地以降に作成して実行した時のフラグの変化を調べなさい。

(1)

```
        ORG 8000H
        LD A,0FCH
    L1:
        INC A
        JR NZ,L1
        HALT
        END
```

(2)

```
        ORG 8000H
        LD A,00H
```

```
        ADD A,A
        LD A,02AH
        XOR 0FFH
        XOR 0FFH
        HALT
        END
```

課題５

次のプログラムを実行し、スタック・ポインタ　スタックの内容の変化を調べなさい。

(1)

```
        ORG 8000H
        LD SP,8200H
        LD B,12H
        LD C,34H
        PUSH BC
        LD D,56H
        LD E,78H
        PUSH DE
        POP HL
        POP DE
        HALT
        END
```

上記のプログラムを機械語に変換し、Z 80 のステップモードを用いて各レジスター、メモリの内容を確認しなさい。

(2)

```
        ORG 8000H
        LD SP,8200H
        LD B,12H
        LD C,34H
        CALL L2
        LD (8500H),a
        HALT
```

```
        ORG 8400H
    L2:
        LD A,B
        ADD A,C
        RET
        END
```

上記のプログラムを機械語に変換し、Z 80 のステップモードを用いて各レジスター、メモリの内容を確認しなさい。

課題５　（１）

プログラム	機械語	各レジスターの内容							各スタックの内容			
		B	C	D	E	H	L	SP	81FFH	81FEH	81FDH	81FCH
START:LD SP,8200H												
LD B,12H												
LD C,34H												
PUSH BC												
LD D,56H												
LD E,78H												
PUSH DE												
POP HL												
POP DE												
RET												

課題５　（２）

プログラム	機械語	各レジスターの内容							各メモリーの内容		
		A	B	C	D	E	L	SP	8500H	81FEH	81FFH
START:LD SP,8200H											
LD B,12H											
LD C,34H											
CALL L2											
LD (8500H),A											
HALT											
ORG 8400H											
L2:											
LD A,B											
ADD A,C											
RET											
END											

課題６

この課題でマイクロコンピュータの実習装置に LED 表示器を接続します。まずこのプログラムの概要を以下に説明する。LED で表示するため I/O ポートを使用する。そのポートアドレス割り当ては次の様になっている。（詳細は LED ボードの取り扱い説明書を参照）

ポート A　アドレス 4H　全ビットを入力に使用。スイッチが接続されている。
ポート B　アドレス 5H　全ビットが出力に使用。LED が接続されている。
ポート C　アドレス 6H　下位４ビットが出力用　LED が接続されている。
　　上位４ビットは入力用　スイッチが接続されている。
コントロールポート　アドレス 7H

実際にプログラムでは実習装置の I/O 機能として 8255 という名称の LSI が使用されており，上記の様に機能させるために，この I/O 用 LSI のコントロールポートに対して 98H という値を書き込む。以下がその部分である。

```
START:
        LD A,98H
        OUT (CT_PORT),A
```

以降では入力用のポートからスイッチの状態を読み取り，出力用のポートへ結果を出力している。なお CPU の処理単位は８ビットであるが，この実習では１２ビットのデータを扱っていることに注意する必要がある。

```
        ORG 8000H

DISPLAY EQU     0040H

PORT_A  EQU     04H
PORT_B  EQU     05H
PORT_C  EQU     06H
CT_PORT EQU     07H

START:
        LD A,98H
        OUT (CT_PORT),A         ; I/O port の初期設定
L1:
        IN      A,(PORT_A)      ; 8 ビットデータの獲得
```

```
OUT     (PORT_B),A      ; 8 ビット出力
IN      A,(PORT_C)      ; 上位 4 ビットにデータ獲得
RRCA                    ; データを右にシフト
RRCA
RRCA
RRCA
AND     0FH             ; 上位 4 ビットをマスク
OUT     (PORT_C),A      ; 出力
JP      L1
END
```

課題 7

7－1

課題 1 と課題 6 のプログラムを参考に、LED ボードの LED を一つだけ点灯し、これを 0.7 秒単位で左にシフトさせるプログラムを作成し実行しなさい。なお一番左の LED が点灯したならば、次は初めの位置に戻しなさい。

7－2

上記のプログラムで一番左の LED が点灯したならば、次は右にシフトし最初の位置に戻ったら右にシフトを始めるように変更しなさい。

課題 8

課題 7 までに使用した LED ボードやスイッチを利用して 7-1 や 7-2 のプログラムを改造してなにかの動作をさせ，そのプログラムの動作の説明をしなさい。

参考文献

[1]“ Z80 アセンブラ入門 ”，堀桂太郎，浅川毅，東京電機大学出版，2004

[2]“ Microcomputer Components Data book ”，Zilog February,1980

[3]“ Z-80 プログラミング ”，大下真二郎，学献社, 1984

[4]“ はじめて読む Pentium －－マシン語入門編－－ ”，蒲地輝尚，水越康博，ASCII，2004