マイクロプロセッサ レポート

高砂 爽

2023 年 5 月 15 日

1　目的

本実験ではマイクロプロセッサを用いて以下の内容を目的とする．

- 2 進数，10 進数，16 進数の相互変換と補数を用いた数値の表現
- アセンブリ言語を用いたプログラミング，手作業によるアセンブル

2　実験原理・器材（KUE-CHIP2 について）

KUE-CHIP2(Kyoto University Education-CHIP2) とは，教育用の 8 ビットマイクロプロセッサであり，

- 京都高度技術研究所
- 九州大学
- 京都大学
- 立命館大学

によって開発された計算機である．KUE-CHIP2 教育用ボードには，観測や実験のために，スイッチ，表示用 LED，コネクタを揃えているほか，KUE-CHIP2 以外にもユーザーインターフェイスのために，多くの IC を搭載している KUE-CHIP2 の特徴として，単純な命令セット・アーキテクチャを持っている．また，動作が理解しやすいように，内部レジスタの値を観測・制御できたり，1 命令・1 フェーズごとに動作させたりできるという特徴がある．

3　Problem 3.1 加算プログラムのトレース

3.1　実験課題 (方法)

この課題では，すでにアセンブリ言語で書かれたプログラムと機械語が与えられている．これらを利用して，KUE-CHIP2 に加算プログラムを記述し，KUE-CHIP2 の基本的な使い方を学ぶ．

3.2　与えられた式のトレース結果の表

本節では以下の 6 つの式を用いて実験を行った．

式 (1) $2 + 3$
式 (2) $126 + 1$
式 (3) $126 + 2$
式 (4) $2 + (-3)$
式 (5) $(-127) + (-1)$
式 (6) $(-127) + (-2)$

表 1 は，式 (1) を実行した時の各トレース結果である．

また，表 2，表 3 は式 (2) から式 (6) の FLAG のみの値をトレースしている．表 2 では和を計算する時に ADD 命令を，表 3 では和を計算する時に ADC 命令を使用している．なお，これらの表の値はすべて 16 進数を用いて表現している．

表 1: 2 + 3 を行った時の各トレース結果

	PC	FLAG	ACC	IX	DBi	DBo	MAR	IR
	00	00	00	00	64	64	00	00
LD ACC, [D1] P0	↓							
	01	00	00	00	64	64	00	00
LD ACC, [D1] P1								
	01	00	00	00	64	64	00	64
LD ACC, [D1] P2								
	02	00	00	00	80	80	01	64
LD ACC, [D1] P3								
	02	00	00	00	02	02	80	64
LD ACC, [D1] P4								
	02	00	02	00	02	02	80	64
ADD ACC, [D2] P0								
	03	00	02	00	B4	B4	02	64
ADD ACC, [D2] P1								
	03	00	02	00	B4	B4	02	B4
ADD ACC, [D2] P2								
	04	00	02	00	81	81	03	B4
ADD ACC, [D2] P3								
	04	00	02	00	03	05	81	B4
ADD ACC, [D2] P4								
	04	00	05	00	03	03	81	B4
ST ACC, [ANS] P0								
	05	00	05	00	74	74	04	B4
ST ACC, [ANS] P1								
	05	00	05	00	74	74	04	74
ST ACC, [ANS] P2								
	06	00	05	00	82	82	05	74
ST ACC, [ANS] P3								
	06	00	05	00	05	05	82	74
ST ACC, [ANS] P4								
	06	00	05	00	05	05	82	74
HLT P0								
	07	00	05	00	0F	0F	06	74
HLT P1								
	07	00	05	00	0F	0F	06	0F
HLT P2								
	07	00	05	00	0F	0F	06	0F

表 2: ADD 命令を使用した時の FLAG トレース結果

式	$126+1$	$126+2$	$2+(-3)$	$(-127)+(-1)$	$(-127)+(-2)$
計算結果	7F	80	FF	80	7F
	00	00	00	00	00
ADC ACC, [D2] P0	↓				
	00	00	00	00	00
ADC ACC, [D2] P1					
	00	00	00	00	00
ADC ACC, [D2] P2					
	00	00	00	00	00
ADC ACC, [D2] P3					
	00	00	00	00	00
ADC ACC, [D2] P4					
	00	06	02	02	04

表 3: ADC 命令を使用した時の FLAG トレース結果

式	$126+1$	$126+2$	$2+(-3)$	$(-127)+(-1)$	$(-127)+(-2)$
計算結果	7F	80	FF	80	7F
	00	00	00	00	00
ADC ACC, [D2] P0	↓				
	00	00	00	00	00
ADC ACC, [D2] P1					
	00	00	00	00	00
ADC ACC, [D2] P2					
	00	00	00	00	00
ADC ACC, [D2] P3					
	00	00	00	00	00
ADC ACC, [D2] P4					
	00	06	02	0A	0C

3.3 フェーズ単位の実行中の各レジスタの値の変化について

本節では式 (1) を計算した時の各レジスタの値をフェーズ単位で説明する．
初めに，図 1, 2 を用いて，LD 命令について記述する．

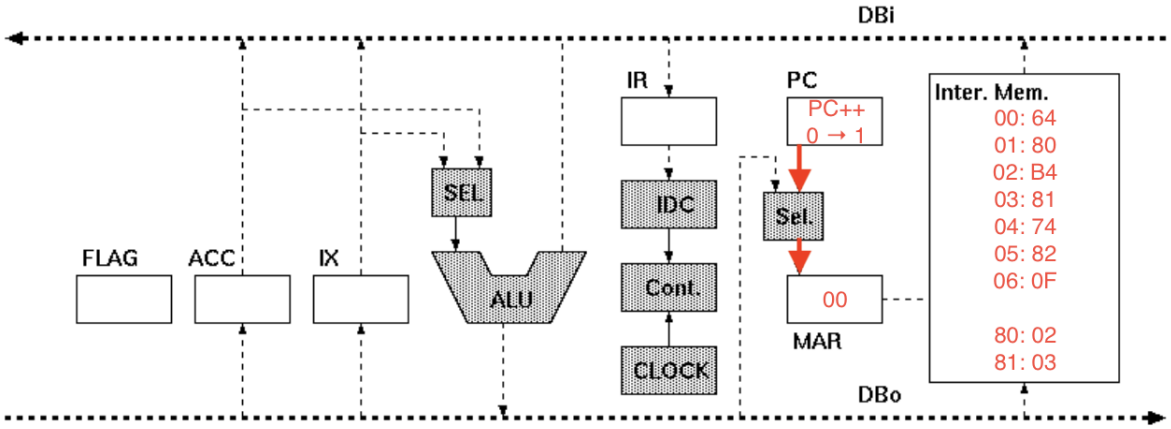
P0　PC の内容”00”を MAR に転送する．
PC が示している 00 番地の命令を読み出すために，MAR に PC の値を転送する．MAR の内容は”00”となり，PC の値はインクリメントされる．

P1　命令内容の読み出しが行われる．
MAR の内容で指定された 00 番地の内容 (64) が入力バス DBi を通り命令レジスタ IR へ転送される．

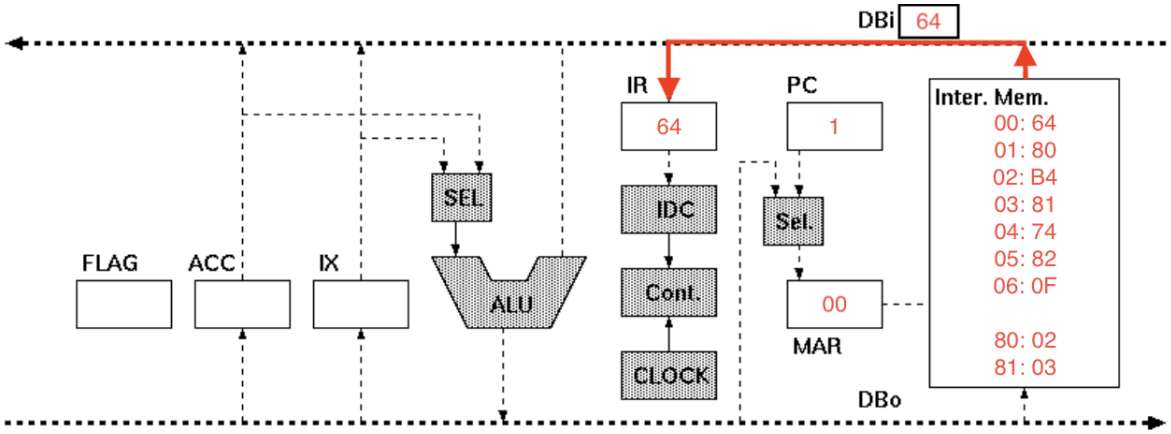
P2　命令デコーダ IDC で IR の内容が解読される．
上位 4 ビット”0110”は LD 命令であること，下位 4 ビット”0100”は，対象としているレジスタが ACC であること，データはプログラム領域にあることが解釈される．
また，第二オペランドのアドレスを生成するために，2 バイト目を読み出す．このため，再び MAR に PC の値”01”を転送し，PC の値はインクリメントされる．

P3　メモリの読み出しが行われる．
MAR の内容で指された 01 番地の内容”80”が読み出され，DBi → ALU → DBo を介して MAR へ転送される．

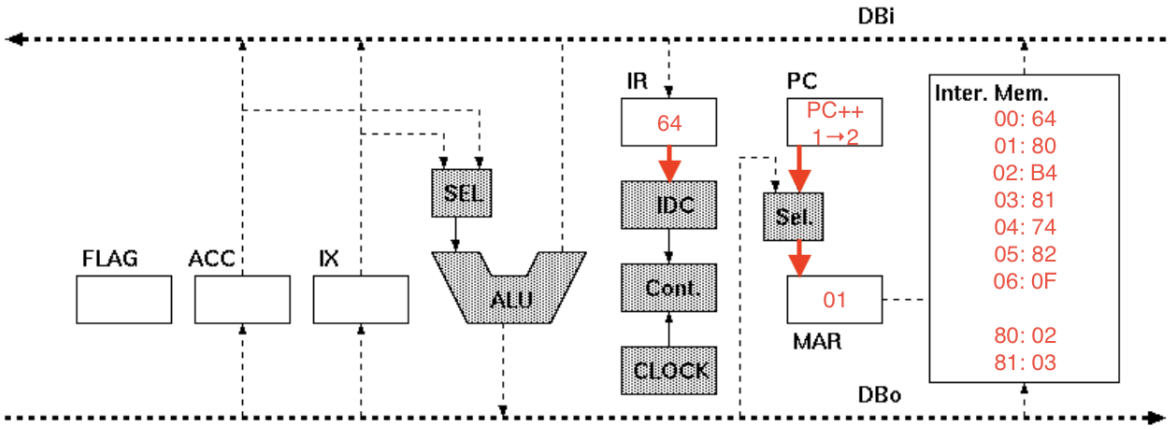
P4　第二オペランドの読み出しが行われる．
命令のアドレスモードはプログラム領域を指しているので，読み出されるデータのアドレスの上位ビットは”0”となる．読み出されたデータは DBi → ALU → DBo を介して，ACC に格納される．



(a) LD ACC, [D1] P0 時の値

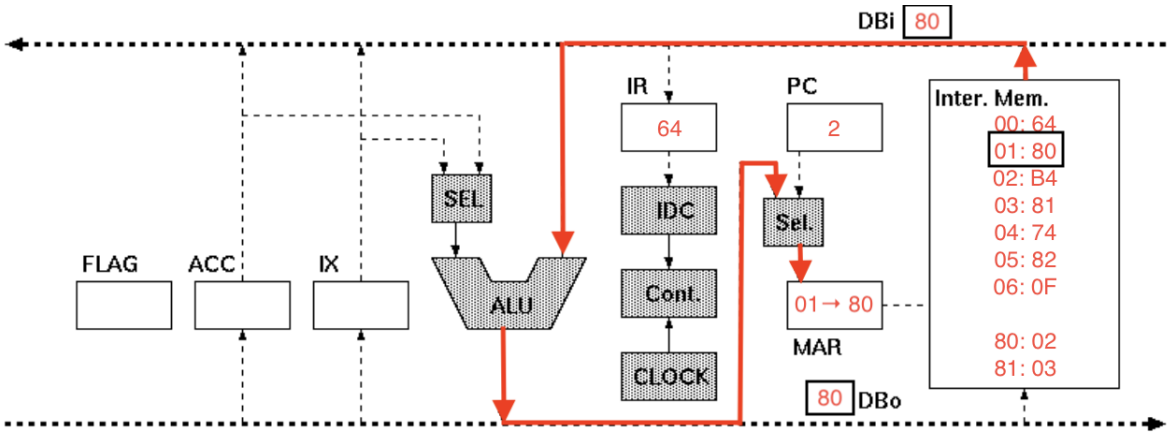


(b) LD ACC, [D1] P1 時の値

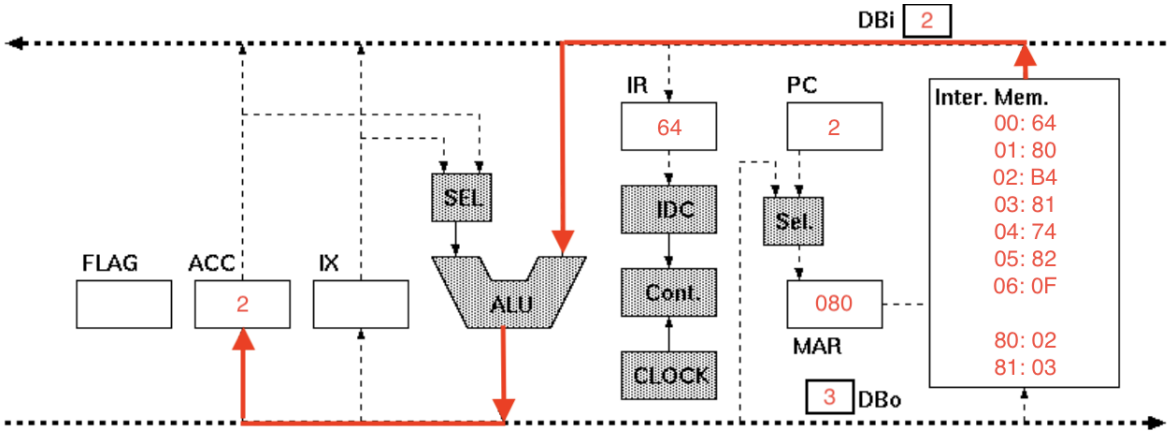


(c) LD ACC, [D1] P2 時の値

図 1: LD ACC 時の各レジスタの値 (1)



(a) LD ACC, [D1] P3 時の値



(b) LD ACC, [D1] P4 時の値

図 2: LD ACC 時の各レジスタの値 (2)

2 つ目に，図 3 を用いて，ADD 命令について記述する．なお，P0, P1 時は全ての命令において共通なので，図は省略する．また，P2 においても省略する．

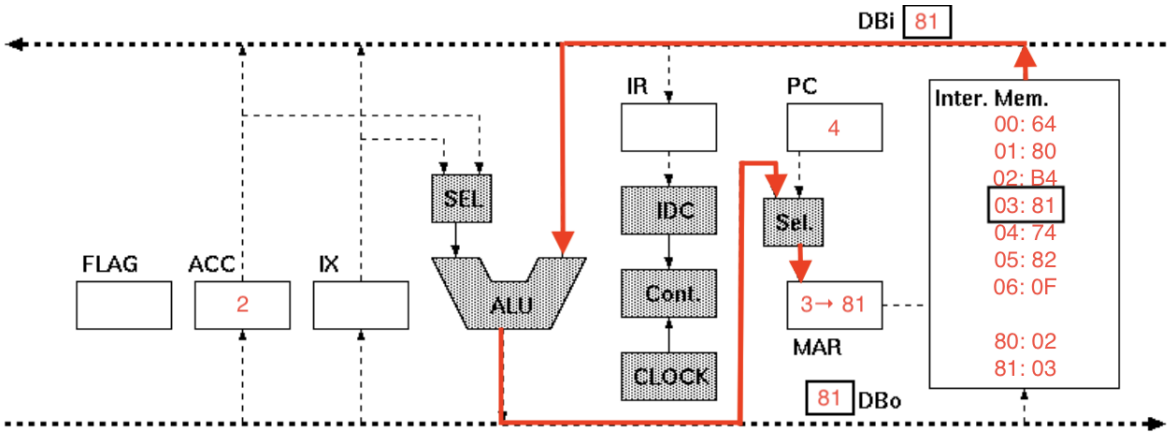
P0　PC の内容”02”を MAR に転送する．
PC が示している 02 番地の命令を読み出すために，MAR に PC の値を転送する．MAR の内容は”02”となり，PC の値はインクリメントされる．

P1　命令内容の読み出しが行われる．
MAR の内容で指定された 02 番地の内容 (B4) が入力バス DBi を通り命令レジスタ IR へ転送される．

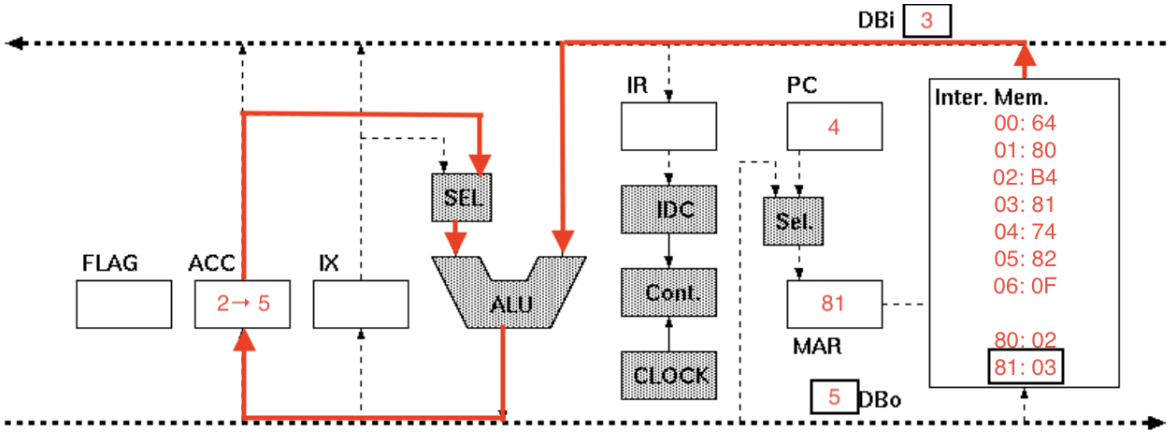
P2　命令でコータ IDC で IR の内容が解読される．
上位 4 ビット”1011”は ADD 命令であること，下位 4 ビット”0100”は，対象としているレジスタが ACC であること，データはプログラム領域にあることが解釈される．
また，第二オペランドのアドレスを生成するために 2 バイト目を読み出す．このため，再び MAR に PC の値”03”を転送し，PC の値はインクリメントされる．

P3　メモリの読み出しが行われる．
MAR の内容で示された 03 番地の内容”81”が読み出され，DBi → ALU → DBo を介して MAR へ転送される．

P4　第二オペランドの読み出しが行われる．
MAR の内容”81”を元に，プログラム領域にある”3”が DBi を介して ALU へ転送される．また，ACC の内容”2”が ALU へ転送され加算，計算結果は DBo を介して ACC に格納される．



(a) ADD ACC, [D2] P3 時の値



(b) ADD ACC, [D2] P4 時の値

図 3: ADD ACC 時の各レジスタの値

3 つ目に，図 4 を用いて，ST 命令について記述する．

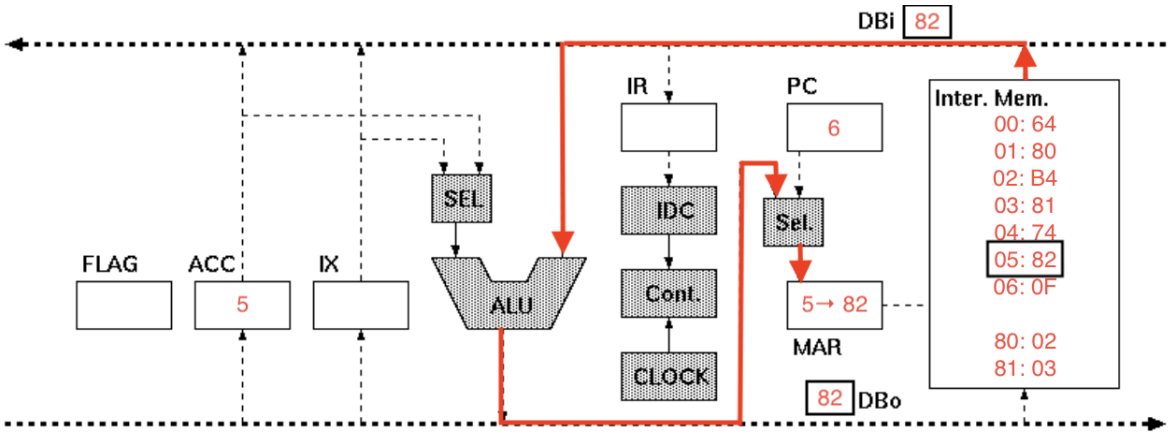
P0　PC の内容”04”を MAR に転送する．
PC が示している 04 番地の命令を読み出すために，MAR に PC の値を転送する．MAR の内容は”04”となり，PC の値はインクリメントされる．

P1　命令内容の読み出しが行われる．
MAR の内容で指定された 04 番地の内容 (74) が入力バス DBi を通り命令レジスタ IR へ転送される．

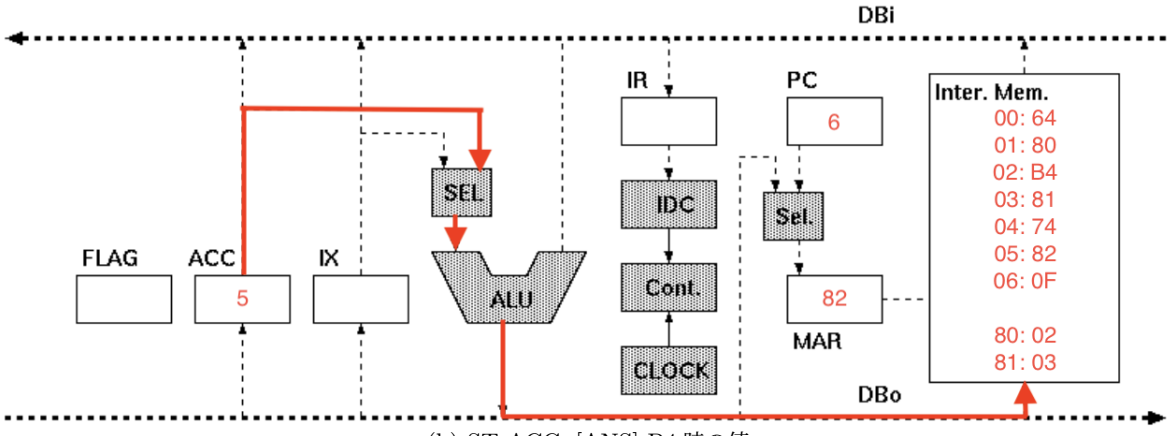
P2　命令でコータ IDC で IR の内容が解読される．
上位 4 ビット”0111”は ST 命令であること，下位 4 ビット”0100”は，対象としているレジスタが ACC であること，データはプログラム領域にあることが解釈される．
また，第二オペランドのアドレスを生成するために 2 バイト目を読み出す．このため，再び MAR に PC の値”05”を転送し，PC の値はインクリメントされる．

P3　メモリの読み出しが行われる．
MAR の内容で示された 05 番地の内容”82”が読み出され，DBi → ALU → DBo を介して MAR へ転送される．

P4　第二オペランドの読み出しが行われる．
命令のアドレスモードはプログラム領域を指しているので，読み出されるデータのアドレスの上位ビットは”0”となる．読み出されたデータは SEL → ALU → DBo を介して，プログラム領域に格納される．



(a) ST ACC, [ANS] P3 時の値



(b) ST ACC, [ANS] P4 時の値

図 4: ST ACC 時の各レジスタの値

最後に，HLT 命令について記述する．

P0　PC の内容”06”を MAR に転送する．
PC が示している 06 番地の命令を読み出すために，MAR に PC の値を転送する．MAR の内容は”06”となり，PC の値はインクリメントされる．

P1　命令内容の読み出しが行われる．
MAR の内容で指定された 06 番地の内容 (0F) が入力バス DBi を通り命令レジスタ IR へ転送される．

P2　命令でコータ IDC で IR の内容が解読される．
”00001- - -”のため，プログラムが停止する．

3.4　各 FLAG がどのような場合に変化するか

本節では，表 2，表 3 を用いて，各 FLAG がどのような場合に変化するかを考察する．

まずはじめに，最終的なフラグレジスタの値は 16 進数で表記されているので，これらを 2 進数表記のなおす．すると，ADD 命令を用いた時の値はそれぞれ，$(00000000)_2$，$(00000110)_2$，$(00000010)_2$，$(00000010)_2$，$(00000100)_2$ となる．次に図 5 のフラグレジスタを参考に値を比較すると，ADD 命令を用いた時は式 (2) ではフラグなし，式 (3) では桁あふれフラグと負フラグ，式 (4)，(5) では負フラグ，式 (6) では桁あふれフラグが立っている．同様に ADC 命令を用いた時は式 (3) では桁あふれフラグと負フラグ，式 (4) では負フラグ，式 (5) では桁あがりフラグと負フラグ，式 (6) では桁あがりフラグと桁あふれフラグが立っている．これらを踏まえて，各フラグがどのような場合に変化するかを考察する．

3.4.1　桁あがりフラグについて

これは，計算結果の桁数が増えた時に立つフラグであると考えられる．式 (5) において，$(-128)_{10}$ というのは $(10000000)_2$ であるから，計算結果が 7 桁から 8 桁に変化していることをフラグで知らせている．また，ADD 命令は使用せず，ADC 命令が使用するフラグであると考察できる．

3.4.2　桁あふれフラグについて

これは，二進数 8 桁 (16 進数 1 桁) で表現しきれない値となった時に立つフラグであると考える．式 (3) において，実際の答えは 128 であるのに対して，計算結果は $(80)_h$ であり，$(-128)_{10}$ である．繰り上がったものが補数と認識され，このような結果になったと考えられる．

3.4.3　負フラグについて

これは，計算結果が負の値の時に立つフラグであると考える．補数を使用するため，その後の処理に問題が起きないよう，フラグを立てることで解決している．

3.4.4　ゼロフラグについて

今回の実験ではこのフラグが立つことはなかったが，これは，計算結果が 0 になった時に立つフラグであると予想できる．

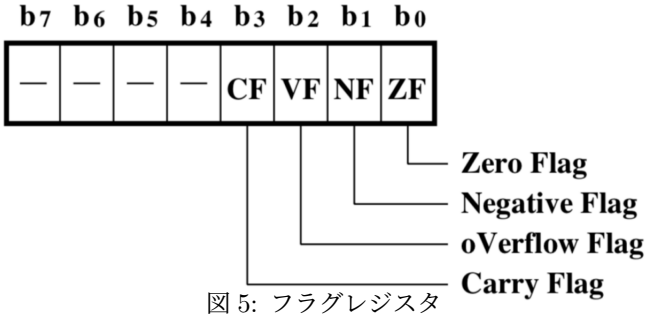


図 5: フラグレジスタ

3.5 ADD と ADC の違い

本節では， 3.4 節を用いて ADD 命令と ADC 命令の違いについて考察する．ADD 命令では桁あがりフラグを使用していなかったのに対し，ADC 命令では桁あがりフラグを使用している．つまり，ADD 命令は半加算機のような動作をし，ADC 命令は全加算機のような動きをすると考えられる．

4 Problem 3.2 乗算プログラムの作成

4.1 実験課題 (方法)

この課題では，フローチャートや，アセンブリ言語を用いたプログラムが与えられていない．そのため，自分で乗算プログラムを作成する．他人のプログラムと比較し，考察を行う．

4.2 フローチャート・プログラムリスト

図 6 は作成したフローチャート図である．また，ソースコード 1 に，本課題にて作成したプログラムリストを記述する．なお，式を与えない状態でのフェーズ数は 56 となった．

なお，被乗数の 1 桁目を A，2 桁目を B とし，乗数の 1 桁目を C，2 桁目を D，答えの 1 桁目を E，2 桁目を F とする．

4.2.1 プログラムの説明

このプログラムは以下の 3 つのフェーズに別れている．

1. E に A の値を C 回繰り返し足す
2. F に B の値を C 回繰り返し足す
3. F に A の値を D 回繰り返し足す

本で，$B \times D$ の計算がなされていないという点については，与えられた問題の前提として，「答えは 3 桁にならない」という条件がある．そのため，$B \times D$ の計算は行っていない．

まず初めに，E と F を 0 で初期化する．初期化を行ったら，次のフェーズに移行する．

1. このフェーズでは，E に A の値を C 回繰り返し足すフェーズである．

1. C が 0 かを確認し，0 でなければ E に A の値を足す．0 であればこのフェーズを終了する．
2. 繰り上がりが起きていれば，F の値をインクリメントする．
3. C の値をデクリメントし，1 番に戻る．

2. このフェーズでは，F に B の値を C 回繰り返し足す．なお，B が 0 であった場合は 0 を B 回足すことになり，非効率である．そのため，B が 0 であった場合は次のフェーズに移行する．また，「答えは 3 桁にならない」という条件から，繰り上がりの処理は行っていない．

1. C が 0 かを確認し，0 でなければ F に B の値を足す．0 であればこのフェーズを終了する．
2. C の値をデクリメントし，1 番に戻る．

3. このフェーズでは，F に A の値を D 回繰り返し足す．

1. D が 0 かを確認し，0 でなければ F に A の値を足す．0 であればこのフェーズを終了する．
2. D の値をデクリメントし，1 番に戻る．

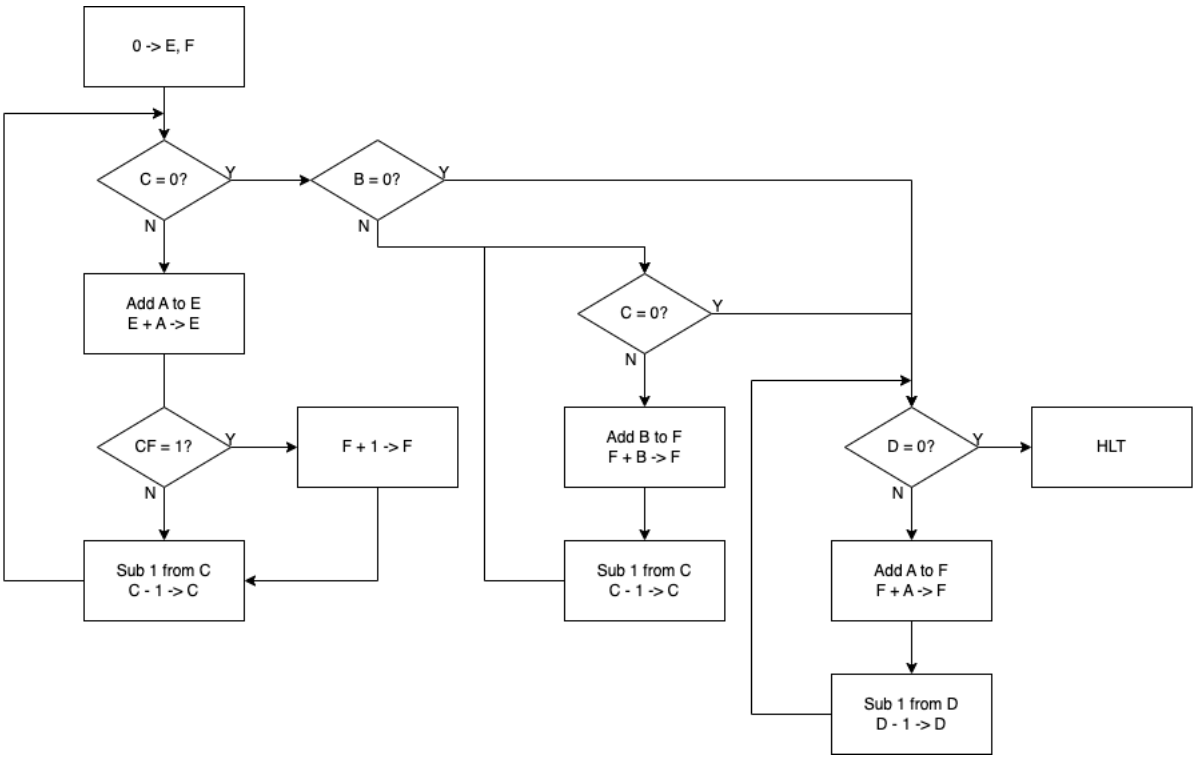


図 6: 乗算プログラムのフローチャート

ソースコード 1: 乗算を行うプログラムリスト

```
80  :          A:      EQU     80H
81  :          B:      EQU     81H
82  :          C:      EQU     82H
83  :          D:      EQU     83H
```

```
84 :                    E:          EQU             84H
85 :                    F:          EQU             85H
00 :    62 00                       LD      ACC,    0
02 :    74 84                       ST      ACC,    [E]
04 :    74 85                       ST      ACC,    [F]
06 :                    TOFIRST:
06 :    6C 82                       LD      IX,     [C]
08 :    AA 00                       SUB     IX,     0
0A :    39 23                       BZ              TOSECOND
0C :    30 0E                       BA              LOOP1
0E :                    LOOP1:
0E :    20                          RCF
0F :    64 84                       LD      ACC,    [E]
11 :    94 80                       ADC     ACC,    [A]
13 :    74 84                       ST      ACC,    [E]
15 :    35 1D                       BNC             PASS1
17 :    64 85                       LD      ACC,    [F]
19 :    B2 01                       ADD     ACC,    1
1B :    74 85                       ST      ACC,    [F]
1D :                    PASS1:
1D :    AA 01                       SUB     IX,     1
1F :    39 23                       BZ              TOSECOND
21 :    30 0E                       BA              LOOP1
23 :                    TOSECOND:
23 :    64 81                       LD      ACC,    [B]
25 :    F2 00                       CMP     ACC,    0
27 :    39 3E                       BZ              TOTHERD
29 :    6C 82                       LD      IX,     [C]
2B :    AA 00                       SUB     IX,     0
2D :    39 3E                       BZ              TOTHERD
2F :    30 31                       BA              LOOP2
31 :                    LOOP2:
31 :    20                          RCF
32 :    64 85                       LD      ACC,    [F]
34 :    B4 81                       ADD     ACC,    [B]
36 :    74 85                       ST      ACC,    [F]
38 :    AA 01                       SUB     IX,     1
```

```
3A :  39 3E                BZ          TOTHERD
3C :  30 31                BA          LOOP2
3E :                TOTHERD:
3E :  6C 83                LD    IX,   [D]
40 :  AA 00                SUB   IX,   0
42 :  39 53                BZ          EXIT
44 :  30 46                BA          LOOP3
46 :                LOOP3:
46 :  20                   RCF
47 :  64 85                LD    ACC,  [F]
49 :  B4 80                ADD   ACC,  [A]
4B :  74 85                ST    ACC,  [F]
4D :  AA 01                SUB   IX,   1
4F :  39 53                BZ          EXIT
51 :  30 46                BA          LOOP3
53 :                EXIT:
53 :  0F                   HLT
                           END
```

4.3 他人のプログラムとの比較

図 6 を元に作成したプログラムの Byte 数は 88Byte となった．表 4 は実行時間計測用の式である．また，表 5 は各プログラムの Byte 数，実行時間をまとめた表である．なお，実行時間は四捨五入を行っている．

これらの表からわかることとして，3 人の間にプログラムサイズの大きな差はないことがわかる．はじめに，Ⓐ式よりもⒷ式の実行時間が大きい結果となった．考察として，Ⓐ式では，それぞれ $(25)_h$ 回の計算を繰り返しているのに対して，Ⓑ式では $(AB)_h$ 回，$(3)_h$ 回の計算を繰り返している．つまり，繰り返しの量が実行時間に影響を及ぼしていると考えることができる．このプログラムの改善案として，被乗数と乗数の値を比較し，被乗数と乗数を入れ替える操作を行うことで更なる実行時間の短縮が見込める．

結城君のプログラムでは上記の改善案を取り入れて，作成していた．この操作によってⒶ式，Ⓑ式ともに 21 秒という結果になったのだろう．また，末本君のプログラムでは，シフト演算を用いて乗算を行っていた．シフト演算を用いた乗算では，乗数のビット数だけ比較・シフト演算を行うので，足し算の繰り返しよりも実行時間が短い結果となった．

表 4: プログラム計測用の乗算

Ⓐ	03AB
×	0025
	87B7

Ⓑ	0025
×	03AB
	87B7

表 5: 各プログラムの実行時間

名前	使用 Byte 数	Ⓐ式の実行時間	Ⓑ式の実行時間
高砂	88Byte	28 秒	63 秒
結城	74Byte	21 秒	21 秒
末本	89Byte	8 秒	12 秒

4.4 理論値の計算・実測値との比較

本説では，実行時間の計測に使用した 2 式をもとに実行時間の理論値・実測値との比較を行う．今回の実験では，クロックを 100Hz に設定し実験を行った．また，1 クロックで 1 フェーズ進むことから，1 フェーズにかかる実行時間の理論値は 10 ミリ秒であることが考えられる．よって，実行時間の理論値を算出するためには，計算時の総フェーズ数を求めることで理論値が求められると考える．総フェーズ数を求めるため，プログラム 1 を用いて算出する．なお，Ⓐ式の実行時間の実測値は 28 秒，Ⓑ式の実行時間の実測値は 63 秒である．

Ⓐ式では，まず，04 番地にて，ACC の値を [F] に STORE するまで 14 フェーズ使用している．次に，06 番地にて，[C] の値を IX に転送，IX の減算，比較し LOOP1 に行くまでで，17 フェーズ使用している．LOOP1 では，繰り上がりをしない場合は 34 フェーズ，繰り上がりをする場合は 48 フェーズ使用する．今回の式は，37 ループ中，24 回繰り上がりを行うが，最後の足し算のみ分岐にて，TOSECOND に行くので，34 フェーズではなく，30 フェーズとなる．よって，$(34 \times 12) + (48 \times 24) + 30 = 1,590$ フェーズ使用している．次に，23 番地にて，[B] の値を ACC に転送してから，LOOP2 に行くまでで，30 フェーズ使用する．LOOP2 では，一回繰り返すのに，30 フェーズ使用し，繰り上がりを考慮しないが，ここでも，最後の足し算のみ分岐にて，TOTHERD に行くので，30 フェーズではなく，26 フェーズとなる．よって，$30 \times 36 + 26 = 1,106$ フェーズ使用している．次に，3E 番地にて，[D] の値を IX に転送してから，HLT 命令に行くまでで，13 フェーズ使用し，HLT で 3 フェーズ使用し，プログラムが終了する．よって，Ⓐ式を計算する際の総フェーズ数は 2,773 フェーズ，実行時間の理論値は 27.73 秒となる．実測値は 28 秒であったので，概ね良い結果となった．

Ⓑ式でも同様に考えると，LOOP1 までに 31 フェーズ使用し，LOOP1 では，171 ループ中，24 回繰り上がりを行うので，$(34 \times 146) + (48 \times 24) + 30 = 6,146$ フェーズ，LOOP3 までに 30 フェーズ使用し，LOOP3 では，$30 \times 2 + 26 = 86$ フェーズ，HLT に 3 フェーズ使用し，プログラムが終了する．よって，Ⓑ式を計算する際の総フェーズ数は 6,296 フェーズ，実行時間の理論値は 62.96 秒となる．実測値は 63 秒であったので，こちらも同様に良い結果となった．