|  |  |  |  |
| --- | --- | --- | --- |
| 受　付 | 月　　日 | 評　点 |  |

情報・知能工学実験　報告書

|  |  |
| --- | --- |
| 実験題目 | マイクロプロセッサ |

|  |  |  |
| --- | --- | --- |
| 実験　実施日 | 第1回 | 令和６年　　6月　　日 |
| 第2回 | 令和６年　　7月　　日 |
| 第3回 | 令和6年　　7月　　日 |
| 第4回 | 年　　月　　日 |
| 第5回 | 年　　月　　日 |
| 第6回 | 年　　月　　日 |

|  |  |  |
| --- | --- | --- |
|  | 学籍番号 | 氏名 |
| 報告者 | B243343 | 高岡俊輔 |
| 共同　実験者 |  |  |
|  |
|  |
|  |
|  |

|  |  |
| --- | --- |
| 概要 | ­­­ |

1 ⽬的

2 実験原理・器材

3 Problem 3.1 加算プログラムのトレース

3.1 実験課題

以下の①から⑥について計算し，そのトレースを記録する．①については，実行開始から実行終了まで，観測可能なレジスタバスをトレースする．②から⑥については，ADD開始前からADD終了後まで，フラグレジスタのみをトレースする．また，ADD命令をADC命令に変更して，ADC開始前からADC終了後まで，フラグレジスタのみをトレースする．

1. 2 + 3 = ?
2. 126 + 1 = ?
3. 126 + 2 = ?
4. 2 + (-3) = ?
5. -127 + (-1) = ?
6. -127 + (-2) = ?

3.2 式①から⑥のトレース結果

　ADDを用いた式①のトレース結果を表１に示す。

　ADDを用いた式②から⑥のトレース結果を表2に示す。

　ADCを用いた式②から⑥のトレース結果を表3に示す。

表1　式①のトレース結果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | FLAG | ACC | IX | DBi | DBo | MAR | IR |
|  | 00 | 00 | 00 | 00 | 00 | 00 | 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を用いた式②から⑥のトレース結果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | (2) | (3) | (4) | (5) | (6) |
| Result (Hex) | 7F | 80 | FF | 80 | 7F |
|  | 00 | 00 | 00 | 00 | 00 |
| ADD ACC, [D2] P0 | ↓ |  |  |  |  |
|  | 00 | 00 | 00 | 00 | 00 |
| ADD ACC, [D2] P1 |  |  |  |  |  |
|  | 00 | 00 | 00 | 00 | 00 |
| ADD ACC, [D2] P2 |  |  |  |  |  |
|  | 00 | 00 | 00 | 00 | 00 |
| ADD ACC, [D2] P3 |  |  |  |  |  |
|  | 00 | 00 | 00 | 00 | 00 |
| ADD ACC, [D2] P4 |  |  |  |  |  |
|  | 00 | 06 | 02 | 02 | 04 |

表3　ADCを用いた式②から⑥のトレース結果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | (2) | (3) | (4) | (5) | (6) |
| Result (Hex) | 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 考察1

KUE-CHIP2の構成図から，各フェーズのレジスタの値の変化を以下に示す．

表１のトレースから，図1のように初期状態はすべて0の状態で始まる．

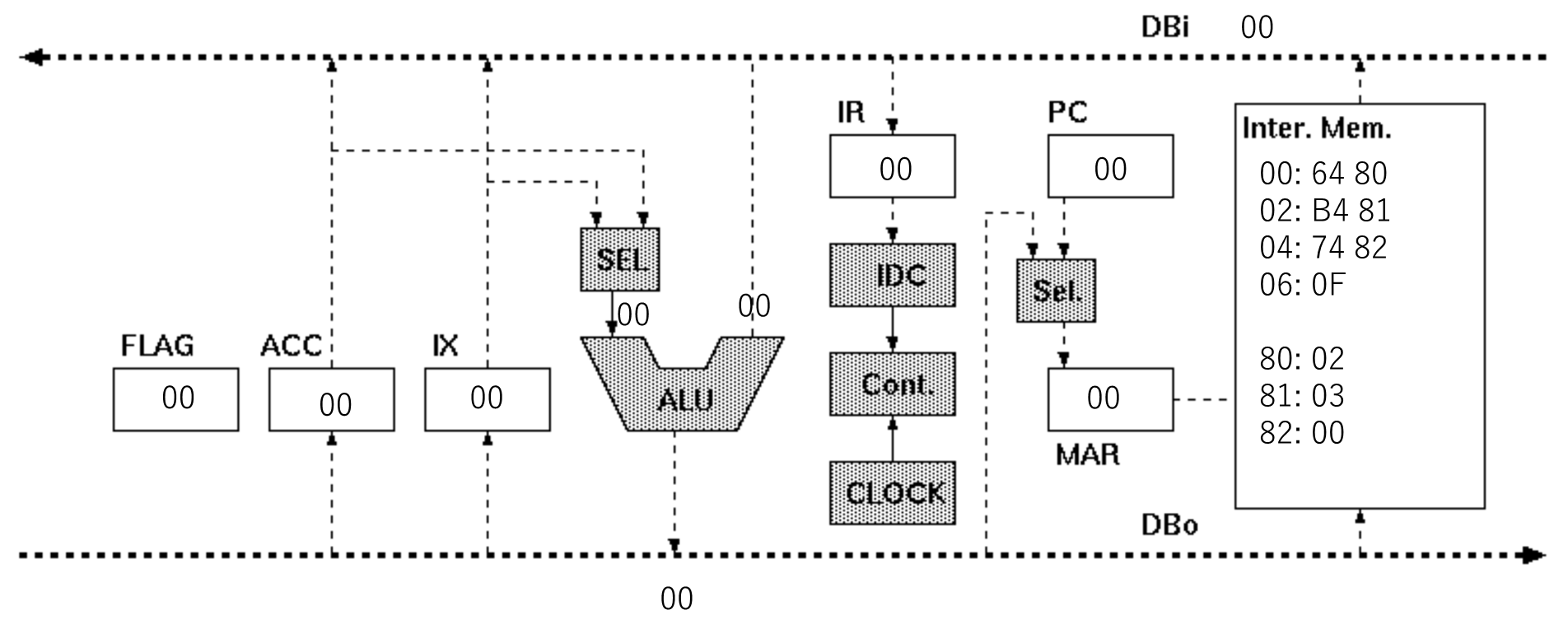


図1 初期状態

　各phaseによって値が移動するときの矢印の色の関係を図2に示す．以降の図はこのルールに沿って示される．また，矢印の横に番号は順序を示す．

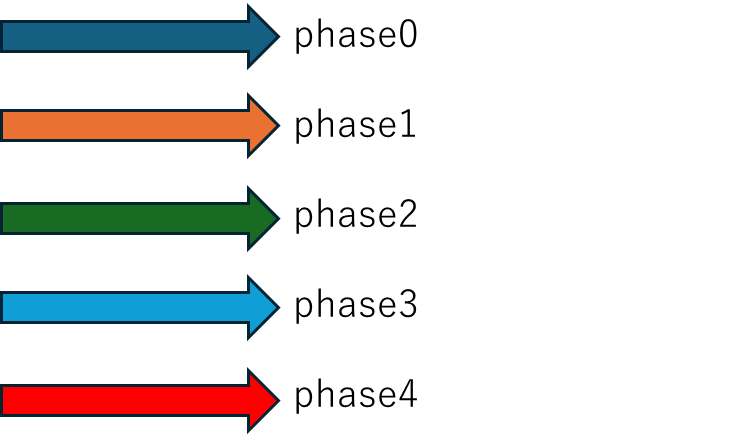


図2 phaseと色の関係

　プログラムが実行されると，まずLD命令が実行される．Phase0では，PCの値がMARに渡され，PCが1加算される．MARが00なので，Memoryの00番地の値がDBiに渡され，DBiの値がALUを経由して，DBoに渡される．Phase1では，DBiの値がIRに渡される．図3にphase0，phase1の様子を示す．

Phase2では，PCの値がMARに渡され，PCが1加算される．MARが01なので，Memoryの01番地の値がDBiに渡され，DBiの値がALUを経由して，DBoに渡される．図4にphase2の様子を示す．

Phase3では，DBoの値をMARに渡する．MARの値が80になったことで，Memoryの80番地の値が，DBiに渡される．また，DBiの値がALUを経由して，DBoに渡される．Phase4では，DBoの値がACCに格納される．図5に，phase3とphase4の様子を示す．これで，LD命令は終了する．

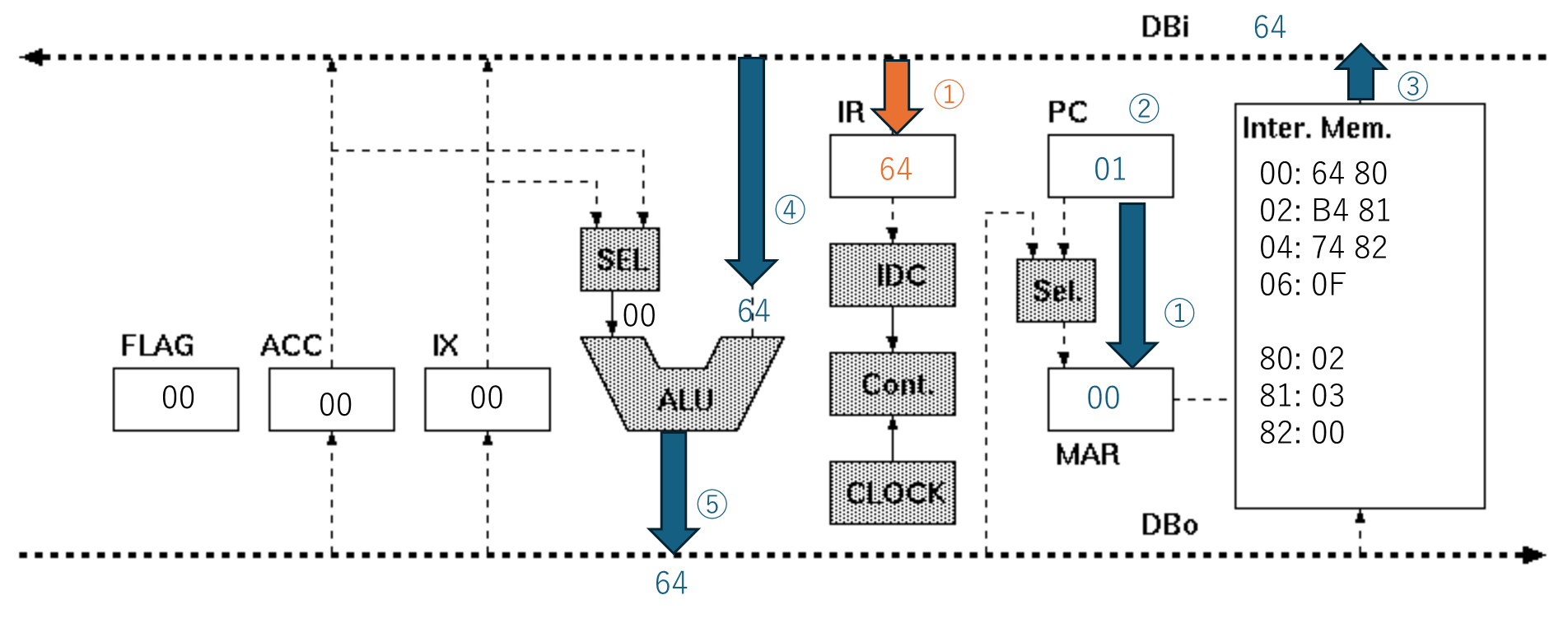


図3 LD命令のphase0，phase1

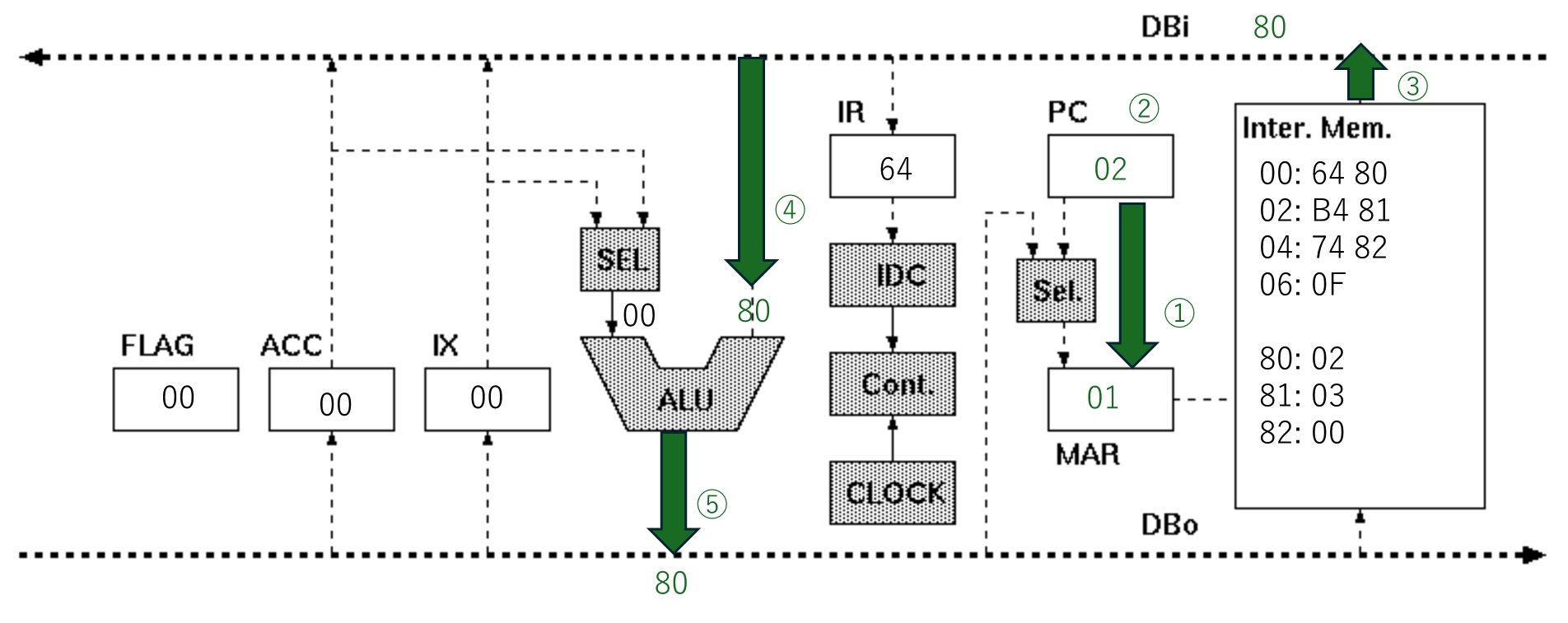


図4 LD命令のphase2

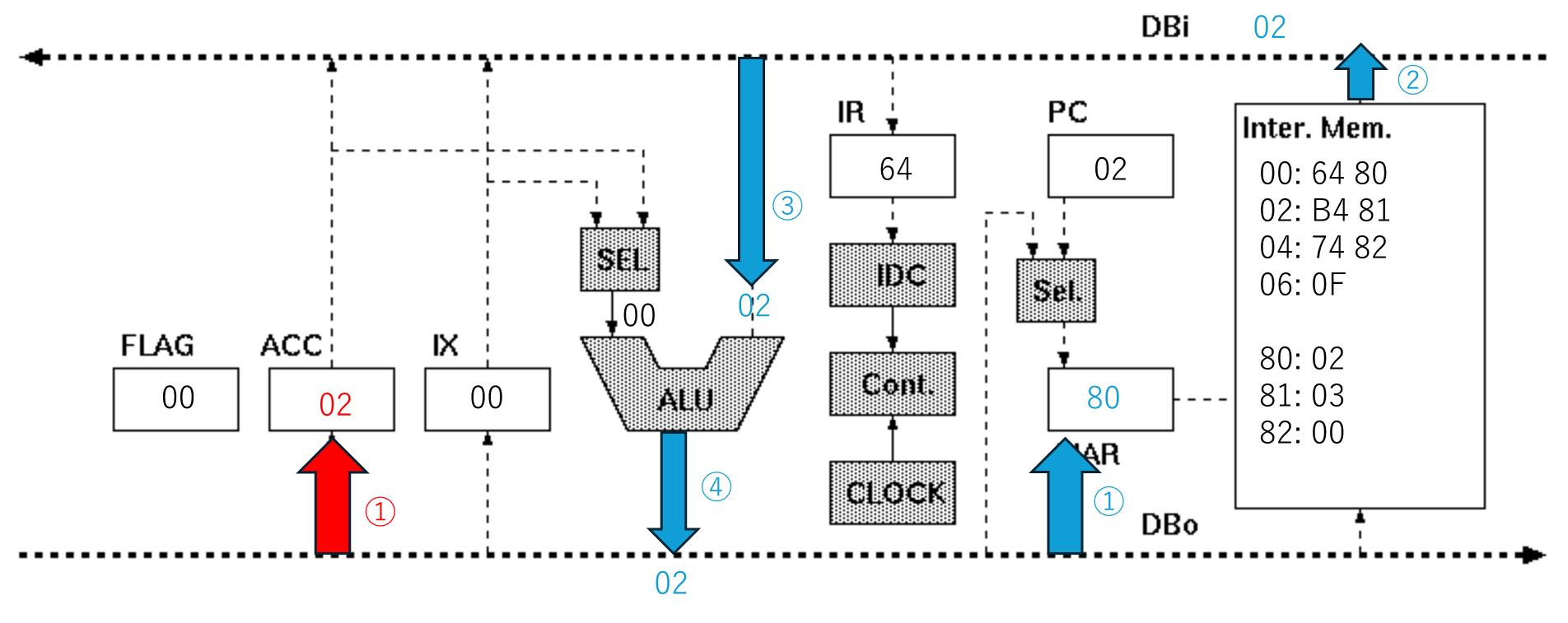


図5 LD命令のphase3，phase4

　次にADD命令が実行される．Phase0では，PCの値がMARに渡され，PCの値が１加算される．MARが02なので，Memoryの02番地の値がDBiに渡され，DBiの値がALUを経由して，DBoに渡される．Phase1では，DBiの値がIRに渡される．図6にphase0，phase1の様子を示す．

Phase2では，PCの値がMARに渡され，PCが1加算される．MARが01なので，Memoryの03番地の値がDBiに渡され，DBiの値がALUを経由して，DBoの渡される．図7にphase2の様子を示す．

Phase3では，DBoの値をMARに渡する．MARの値が81になったことで，Memoryの81番地の値が，DBiに渡される．次に，DBiの値とACCに格納されていた値が，ALUに送られ，加算結果がDBoに渡される．Phase4では，DBoの値がACCに格納される．図8に，phase3とphase4の様子を示す．これで，ADD命令は終了する．

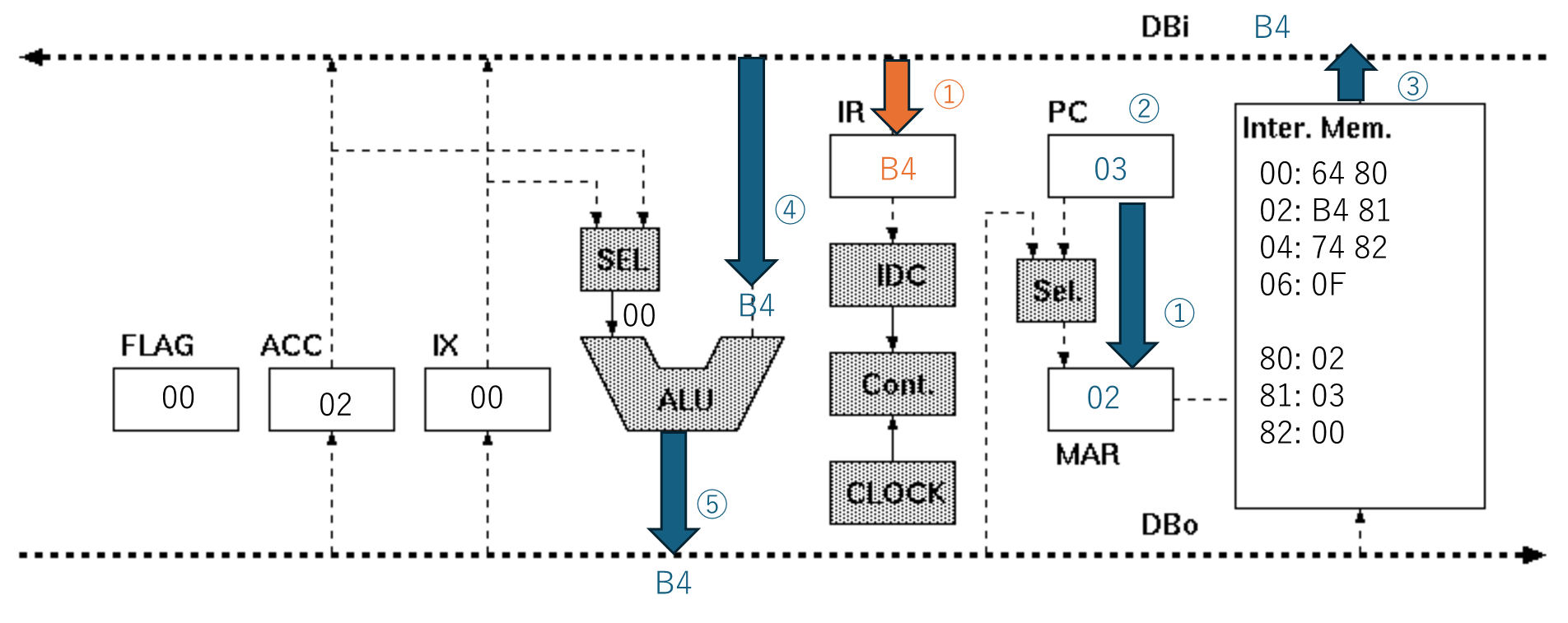


図6 ADD命令のphase0，phase1

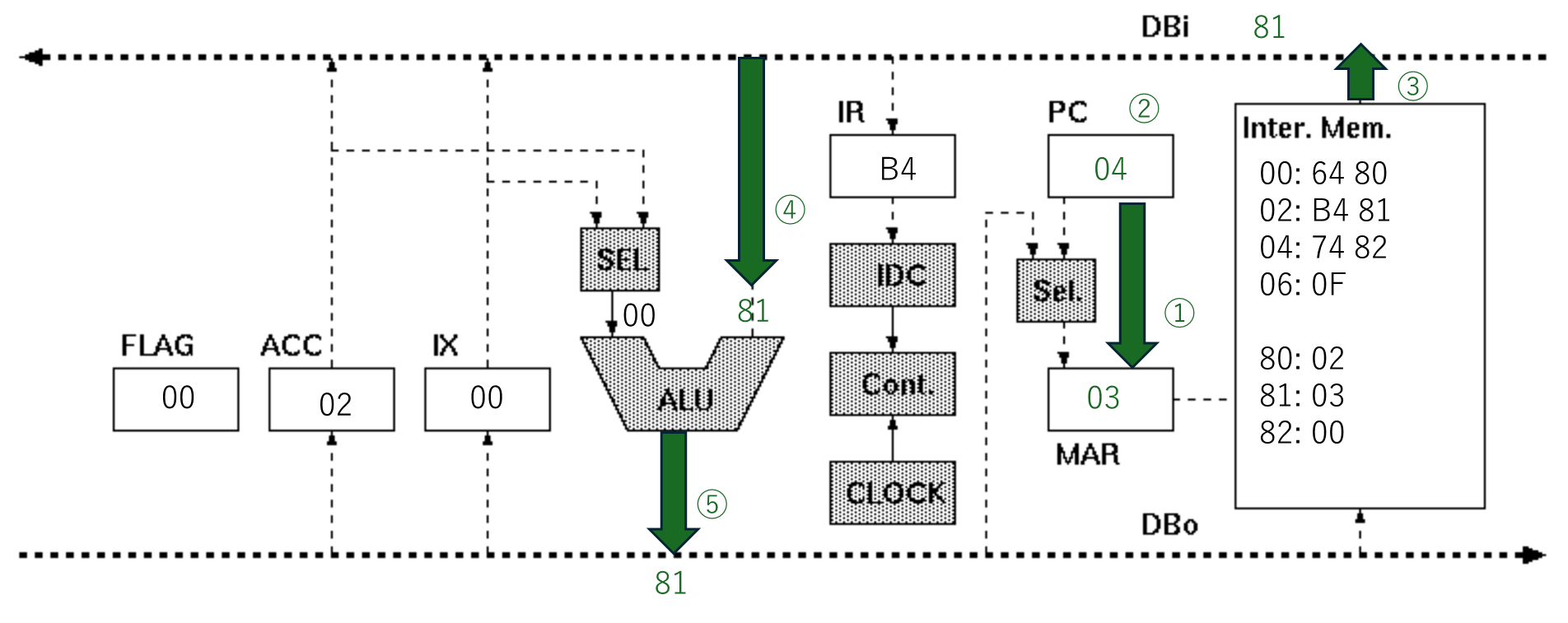


図7 ADD命令のphase2

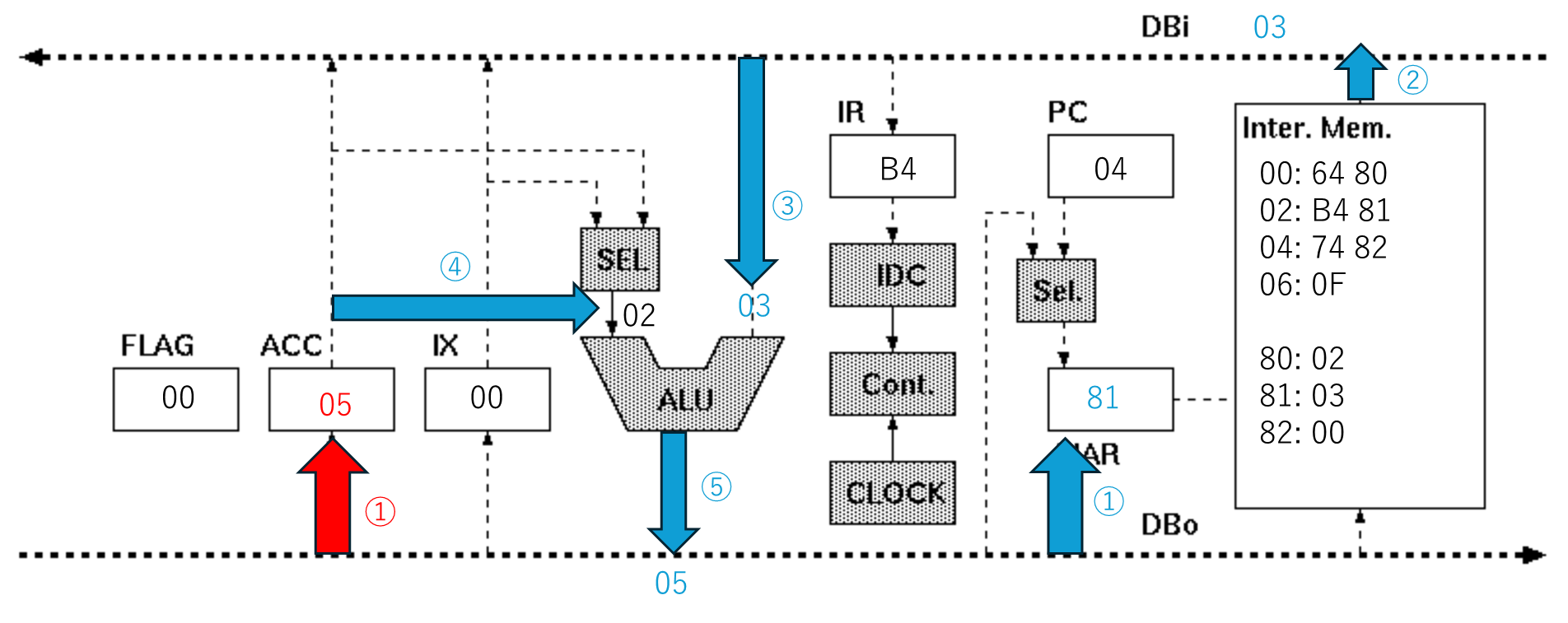


図8 ADD命令のphase3，phase4

　次にST命令が実行される．Phase0では，PCの値がMARに渡され，PCの値が１加算される．MARが04なので，Memoryの04番地の値がDBiに渡され，DBiの値がALUを経由して，DBoに渡される．Phase1では，DBiの値がIRに渡される．図9にphase0，phase1の様子を示す．

Phase2では，PCの値がMARに渡され，PCが1加算される．MARが05なので，Memoryの05番地の値がDBiに渡され，DBiの値がALUを経由して，DBoの渡される．図10にphase2の様子を示す．

Phase3では，DBoの値がMARに渡されMARの値が81になる． また，ACCの値がDBiとDBoにALU経由で渡される．Phase4では，MARの値が82なのでMemoryの82番地にDBoの値が格納される．図11に，phase3とphase4の様子を示す．これで，ST命令は終了する．

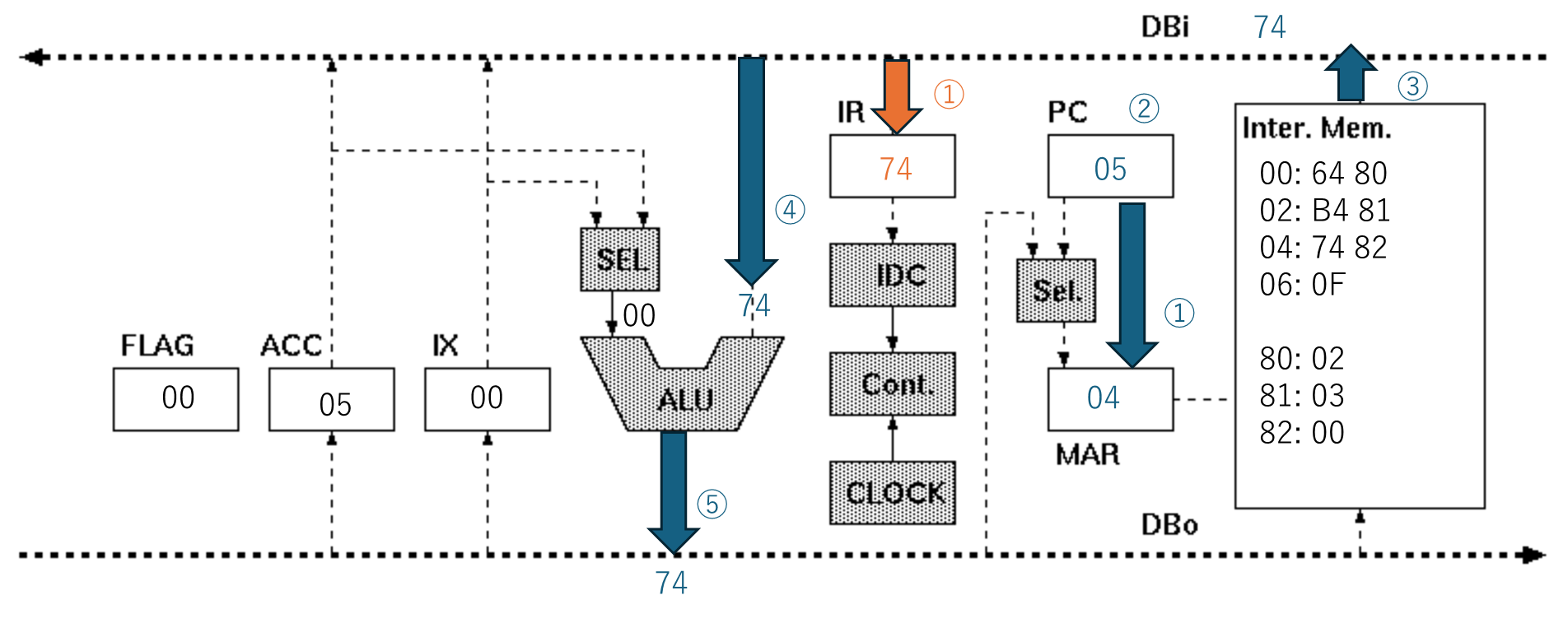


図9 ST命令のphase0，phase1

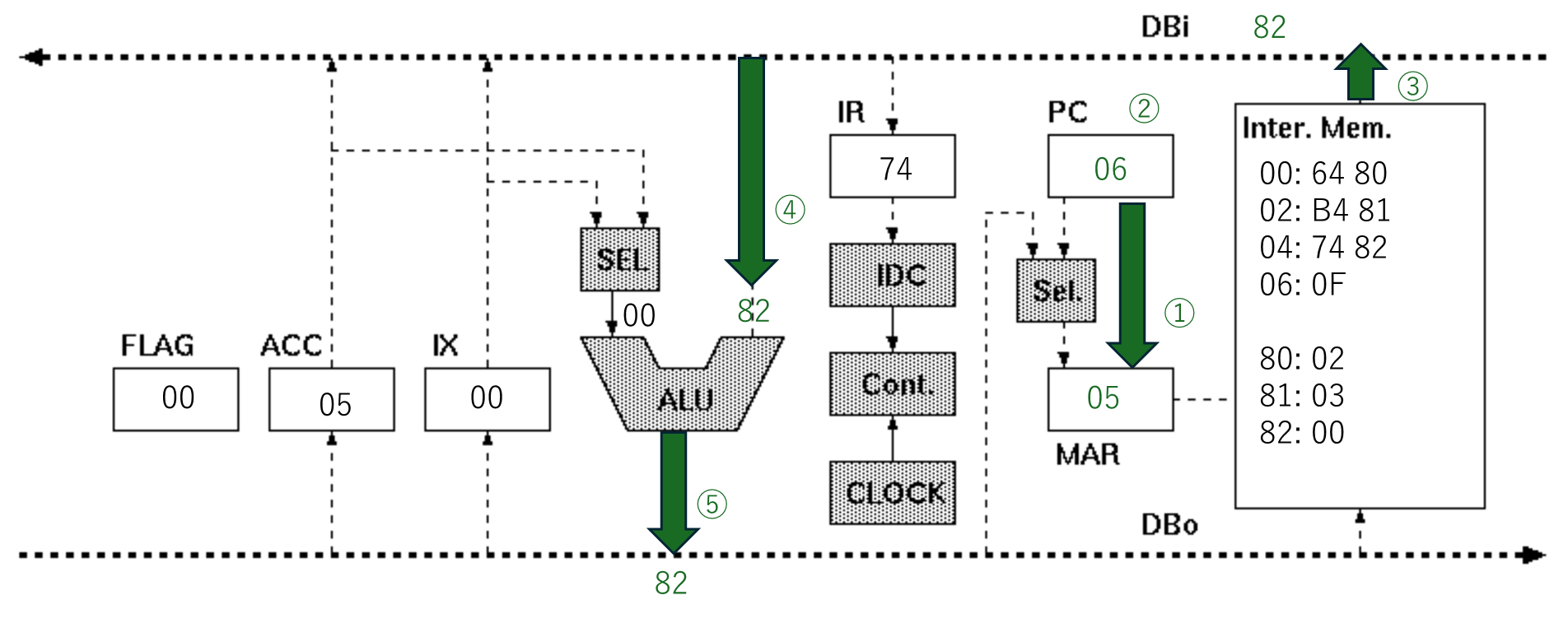


図10 ST命令のphase2

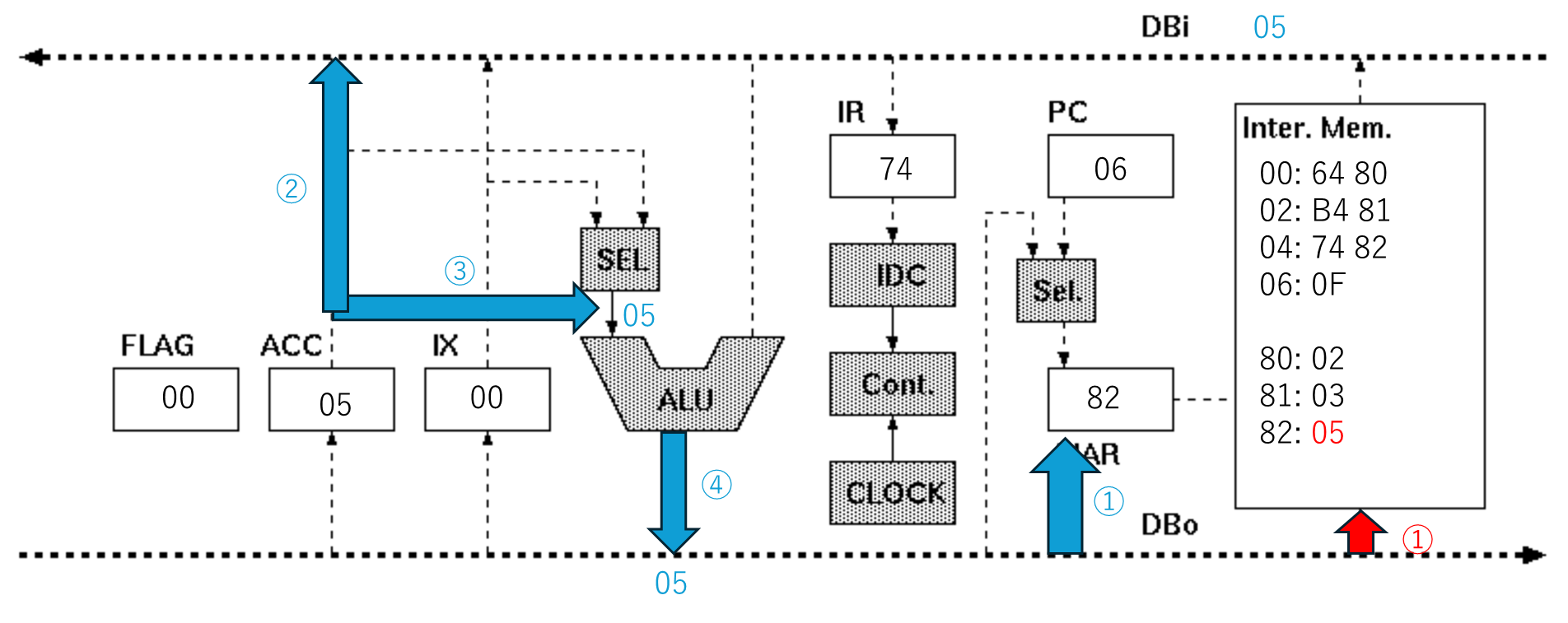


図11 ST命令のphase3, phase4

　　最後に，HLT命令が実行される．Phase0では，PCの値がMARに渡され，PCの値が１加算される．MARが07なので，Memoryの07番地の値がDBiに渡され，DBiの値がALUを経由して，DBoに渡される．Phase1では，DBiの値がIRに渡される．図9にphase0，phase1の様子を示す．

　Phase2では，HALTしプログラムが終了する．

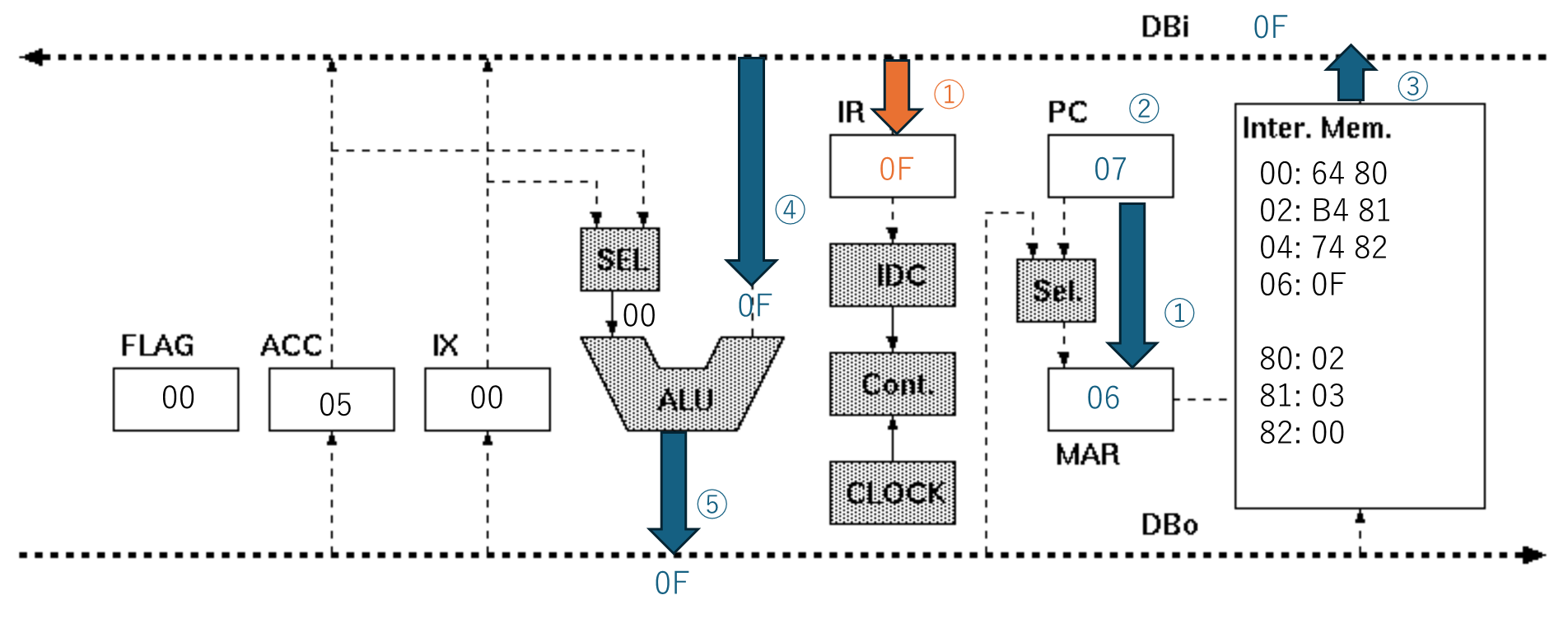


図12 HLT命令のphase0，phase1

3.4 考察２：

　②から⑥のトレース結果から，フラグレジスタがどのような場合に変化するのか以下に示す．図１は指導書より，KUE-CHIP2のフラグレジスタの構成である．

ダイアグラム

自動的に生成された説明

図13 フラグレジスタ

表3,4について，計算方法，立ち上がったフラグを表４にまとめた．

表4 ②から⑥の計算方法と立ち上がったフラグ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| ADD | 00 | 06 | 02 | 02 | 04 |
| ADC | 00 | 06 | 02 | 0A | 0C |

　②の計算は，126 + 1=127である．２進数にすると，0111 1110 + 0000 0001 = 0111 1111である．この計算によってADD，ADCにおいてフラグは上がっていない．

　③の計算は，126 + 2 = 128である．２進数にすると，0111 1110 + 0000 0010 = 1000 0000である．この計算によって，ADD，ADCにおいてフラグは06となっている．図13より，フラグが06のときVFとNFが上がっていることになる．これは，符号付２進数で考えたとき，計算結果である1000 0000は-128になるため，正の数を足した結果が負になるのはおかしいのでオーバーフローしている．よって，VFが立つと考えられる．また，計算結果が負に見えることから，NFが立つと考えられる．

　④の計算は，2 + ( -3 ) = -1 である．２進数にすると，0000 0010 + 1111 1101 = 1111 1111である．この計算によって，ADD，ADCにおいてフラグは02となっている．図13より，フラグが02のときNFのみが上がっていることになる．これは，符号付２進数で考えたとき，計算結果である1111 1111が-1になるため，負の数になっているため，NFが立つと考えられる．

⑤の計算は，-127 + (-1) = -128である．２進数にすると，1000 0001 + 1111 1111 = 1000 0000である．この計算によって，ADDにおいては，02のフラグが立ちこれは，NFを表す．ADCにおいては，0Aのフラグが立ちこれは，CFとNFが立っていることを示す．ADD, ADCのNFは，計算結果を符号付２進数で見ると，1000 0000は-128であるため，負の数である．よって，NFが立つと考えられる．ADCのCFは，計算を9bitで見るとわかる．式は0 1000 0001 + 0 1111 1111 = 1 1000 0000となる．よって，計算結果の9bit目が繰り上がって1になっているので，CFが立つと考えられる．

⑥の計算は，-127 + (-2) = -129である．２進数にすると，1000 0001 + 1111 1110 = 0111 1111である．この計算によって，ADDにおいては，04のフラグが立ちこれは，VFを表す．ADCにおいては，0Cのフラグが立ちこれは，CFとVFが立っていることを示す．ADD，ADCにおいて，VFが立っている．計算結果を符号付２進数で見ると0111 1111は127である．負の数を足した結果が正の数になるのはおかしいのでオーバーフローしている．よって，VFが立つと考えられる．ADCにおいて，CFが立っている．これは，計算を9bitで考えるとわかる．式は，1 1000 0001 + 1 1111 1111 = 1 0111 1111となる．9bit目が繰り上がり1となっているため，CFが立つと考えられる．

　以上のことから，整理すると，CFはADCを用いた計算において，9bit目への繰り上げがある計算の場合に立つことが分かる．VFはADD，ADCにおいて，式と計算結果に矛盾が生じるオーバーフローが生じたときに立つことが分かる．NFは，ADD，ADCにおいて，計算結果を符号付２進数で見たとき，負の値のとき立つことがわかった．

3.5 考察３： ADD と ADC の違いを説明する

　考察2より，計算結果において，繰り上がりが生じた際にCFが立つのがADCである．

　指導書より命令セットの一部を抜粋したものを図14に示す．これを見ると，ADDを用いる場合は，単純に2つの数の加算を行うが，ADCを用いる場合は，2つの数に加え，CFを含めて加算を行うことが分かる．



図14 命令セットの一部抜粋

4 Problem 3.2 乗算プログラムの作成

4.1 実験課題

　符号なし2バイト精度の２数の積を求める乗算プログラムを製作すること．

4.2 フローチャート，プログラムリスト（フェーズ数も記載）を記載し，⽂章でプログラムの説明をする

　以下に作成した乗算プログラムを示す．

|  |
| --- |
| #乗数  M1: EQU 81H  M2: EQU 80H  #被乗数  Q1: EQU 83H  Q2: EQU 82H  P1: EQU 85H  P2: EQU 84H  n: EQU 86H  LD ACC, 00H  ST ACC, [P1]  ST ACC, [P2]  LD ACC, 10H  ST ACC, [n]  LOOP: LD ACC, [Q1]  AND ACC, 01H  CMP ACC, 00H  BZ SHIFT  RCF  LD ACC, [P1]  ADC ACC, [M1]  ST ACC, [P1]  LD ACC, [P2]  ADC ACC, [M2]  ST ACC, [P2]  SHIFT:#左←  LD ACC, [M2]  SLL ACC  ST ACC, [M2]  LD ACC, [M1]  SLL ACC  ST ACC, [M1]  BNC SHIFT2  LD ACC, [M2]  ADD ACC, 01H  ST ACC, [M2]  SHIFT2:#右→  LD ACC, [Q1]  SRL ACC  ST ACC, [Q1]    LD ACC, [Q2]  SRL ACC  ST ACC, [Q2]    BNC ND  LD ACC, [Q1]  ADD ACC, 80H  ST ACC, [Q1]  ND:  LD ACC, [n]  SUB ACC, 01H  ST ACC, [n]  BZ END  BA LOOP    END:  HLT  END |

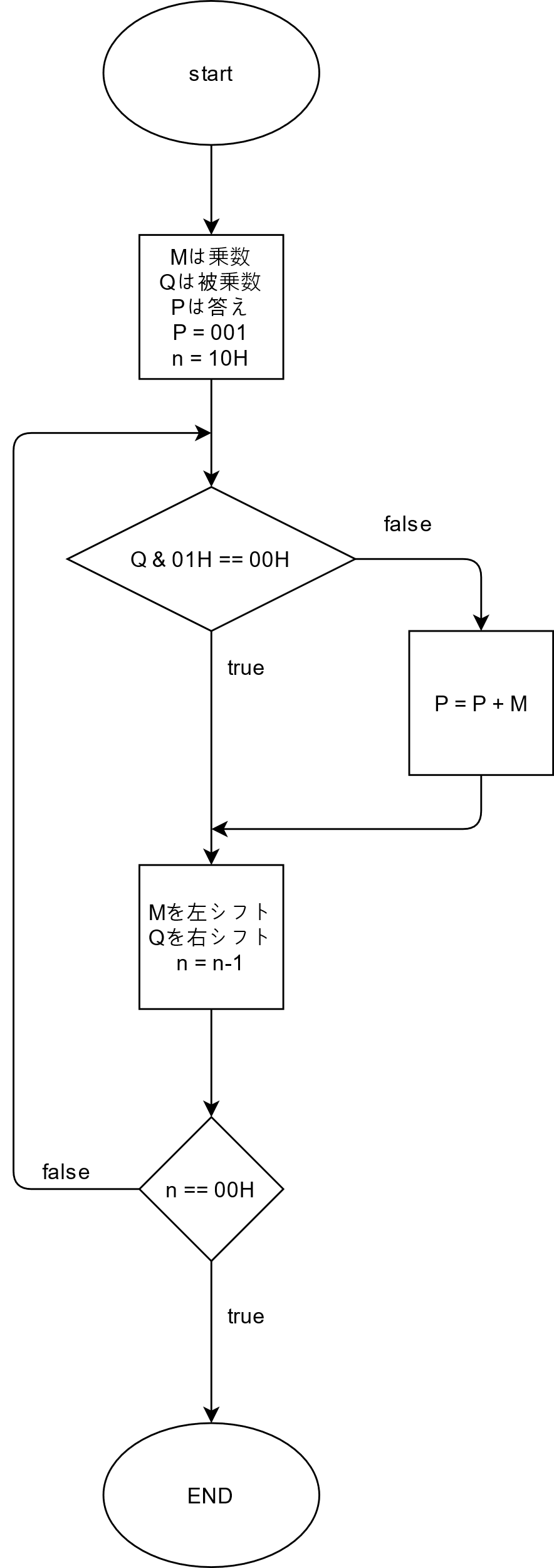


図15 乗算プログラムのフローチャート

　作成したプログラムのフローチャートを図15に示す．作成したプログラムはシフトを用いて乗算を実現した．具体的なステップとしては，

1. 初期設定を行う．
2. 被乗数Qの最下位ビットをチェックし，1なら乗数Mを答えPに加算する．
3. 乗数Mを左シフトし，被乗数Qを右シフトする．
4. ②から③を16回繰り返す．
5. Q\*Mの結果がPに格納されている．

これらのステップによって，乗算を行っている．

4.3 考察 1：⾃分と他の学⽣のプログラムのアルゴリズムを⽐較し，プログラムサイズ，実⾏速度の観点からまとめる

本多幹也

|  |  |  |  |
| --- | --- | --- | --- |
| 本多　幹也 | 53 | 29.33 | 734.08 |

田光佑樹

Aが7.02s Bが12.05sでサイズが54byte

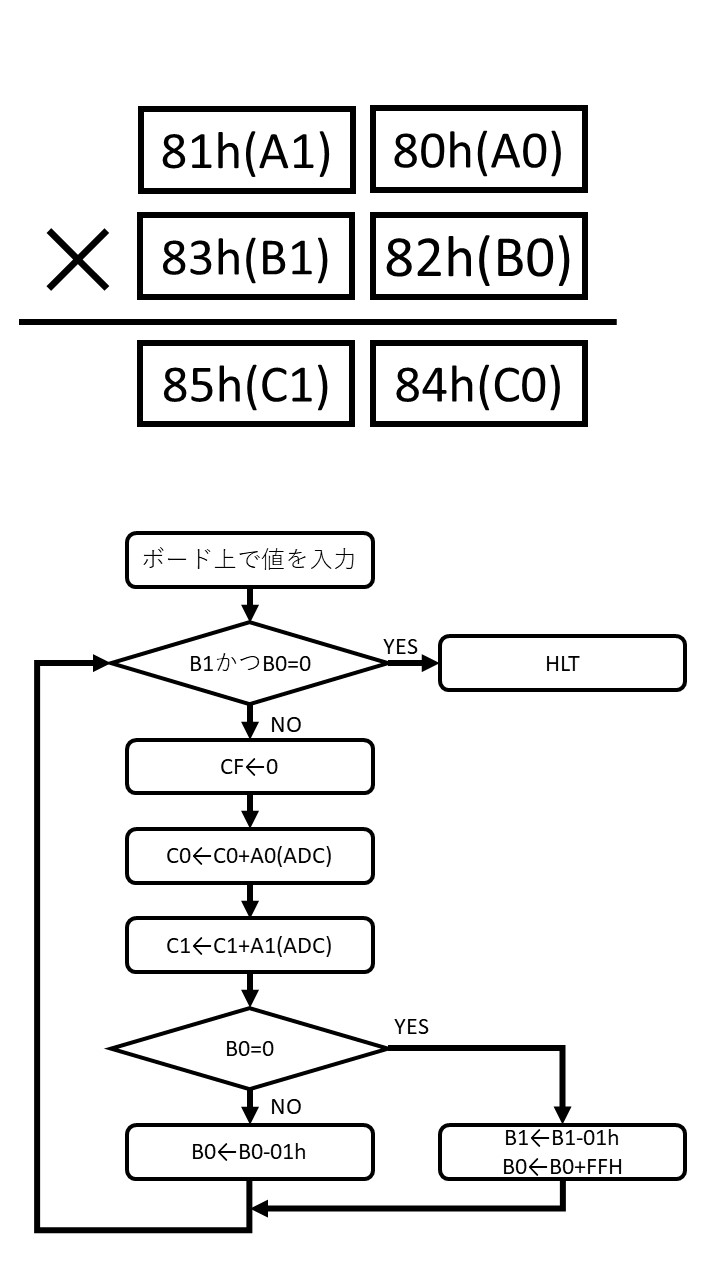


図16　本多幹也の乗算アルゴリズム

4.4 考察 2：Table２を参考に，作成したプログラムの理論値を計算し（計算過

程の説明も記述する），実測値との⽐較を⾏う