# 情報工学実験2

## 「命令フェーズ」

実施日：2010年11月15日


学籍番号：　０９５７０７Ｂ
氏　　名：　大城佳明
締切日：　2010年11月22日
共同実験者：　095701B　青木史林
095703J　岩瀬　翔

# 1　実験目的

機械語 (マシン語) 命令をフェーズ毎に実行させ、そのときのコンピュータ内部の状況を観測することのより、各フェーズでどのような処理が行われているかを調査し、機械語命令の実行の仕組みを理解することを目的とする

# 2　実験概要

KUE-CHIP2 にプログラムを入力し、フェーズの確認を行う。SUB や LD、SCF、AND、BZ 命令を行い、各フェーズを確認することで、機械語 (マシン語) の動作を理解することができる。次に、割り算を行うプログラムを考える。小数以下は切り捨てを行い、割る数か 0 の時は 255 を出力するプログラムを作成する。今までとは違い、実践的な演算のプログラムを考えることにより、より細かく機械語について理解することができる。割り算を行うプログラムを作成し、フェーズについて理解することにより、処理の効率性の考えたプログラムを作成することが可能となる。ここでは効率性の考えたプログラムを作れるかどうかではなく、考えることが大事である。

# 3　実験結果

## 3.1　実験 (1)、(2)、(3) の結果について

命令フェーズを確認するために実行した各アセンブラプログラムとそれに対応するプログラムをすべて示せ。また、必要に応じて、各プログラムを実行する前のレジスタやメモリの初期値も明記せよ。

各アセンブラプログラミングをフェーズ毎に実行したときの実行フェーズ表を全て示せ (実験中に提出した場合は不要)

(実行フェーズの表は実験中に提出したのでここでは省略する。)

### 3.1.1　実験 (1)

1. SUB 命令プログラムを表 1 に示した。

表 1: SUB 命令プログラム

| 番地 | 機械語 | アセンブラ言語 |
|---|---|---|
| 00 | 00 | NOP |
| 01 | A2 | SUB ACC,05H |
| 02 | 05 | |
| 03 | 0F | HLT |

### 3.1.2　実験 (2)

1. LD 命令プログラム (即値) を表 2 に示した。
2. LD 命令プログラム (アドレス) を表 3 に示した。
3. SCF 命令プログラムを表 4 に示した。
4. AND 命令プログラムを表 5 に示した。

表 2: LD 命令プログラム (即値)

| 番地 | 機械語 | アセンブラ言語 |
|---|---|---|
| 00 | 00 | NOP |
| 01 | 62 | LD ACC,05H |
| 02 | 05 | |
| 03 | 0F | HLT |

表 3: LD 命令プログラム (アドレス)

| 番地 | 機械語 | アセンブラ言語 |
|---|---|---|
| 00 | 00 | NOP |
| 01 | 64 | LD ACC,[07H] |
| 02 | 07 | |
| 03 | 0F | HLT |

表 4: SCF 命令プログラム

| 番地 | 機械語 | アセンブラ言語 |
|---|---|---|
| 00 | 00 | NOP |
| 01 | 2F | SCF |
| 02 | 0F | HLT |

表 5: LD 命令プログラム (アドレス)

| 番地 | 機械語 | アセンブラ言語 |
|---|---|---|
| 00 | 00 | NOP |
| 01 | 64 | AND ACC,05H |
| 02 | 05 | |
| 03 | 0F | HLT |

#### 3.1.3 実験 (3)

1. BZ 命令プログラムを表 6 に示した。

表 6: BZ 命令プログラム

| 番地 | 機械語 | アセンブラ言語 |
|---|---|---|
| 00 | AA | SUB IX,01H |
| 01 | 01 | |
| 02 | 39 | BZ jp |
| 03 | 05 | |
| 04 | 0F | HLT |
| 05 | 0F | HLT |

## 3.2 実験 (4) の結果について

各自で作成したアセンブラプログラムと (a)〜(e) のそれぞれの場合の実行結果およびフローチャートを示せ。なお、アセンブラプログラムには、必ず、機械語 (マシン語) プログラムも併記すること。また、必要に応じてプログラムを実行する前のレジスタやメモリの初期値も明記せよ。

各自で作成したアセンブラプログラムがどのような動作をするプログラムなのかを、フローチャートなどを用いて説明し、各実行結果の正当性を示せ。

1. 作成したプログラムを表 7 に示した。
2. プログラムのフローチャートは図 1 に示した。
3. ADD と IX の始めの値は「00H」である。
4. プログラムの説明を行う
   (a) 表 7 の 00,01 番地では ACC に (00H) 番地の値を足し算している。つまり、ACC に (00H) 番地の値を入れている （図 1 の 2）

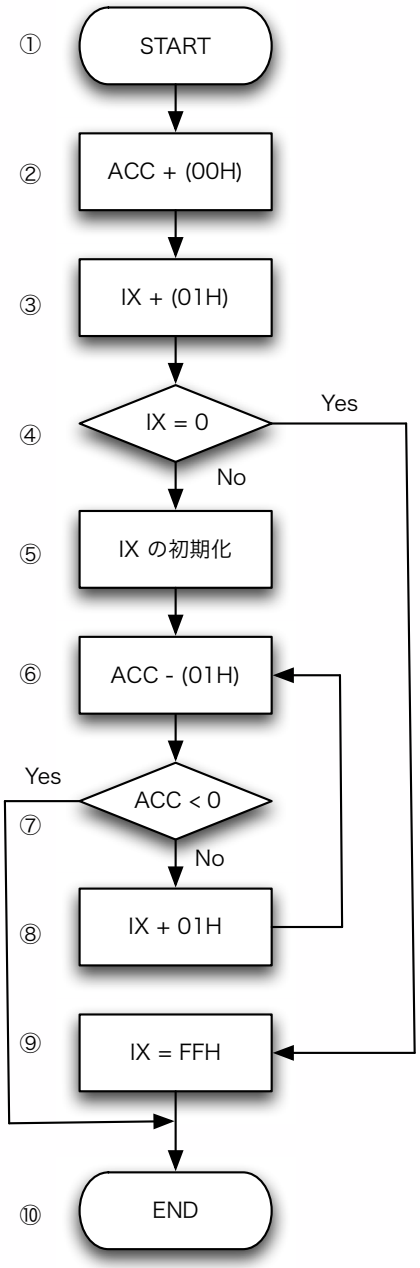


図 1: 割り算プログラム (フローチャート)

表 7: 表のキャプション

| 番地 | 機械語 | アセンブラ言語 |
|---|---|---|
| 00 | B5 | ADD ACC,(00H) |
| 01 | 00 | |
| 02 | BD | ADD IX,(01H) |
| 03 | 01 | |
| 04 | 39 | BZ [0F] |
| 05 | 0F | |
| 06 | C9 | EOR IX.IX |
| 07 | A5 | SUB ACC,01H |
| 08 | 01 | |
| 09 | 3A | BN [11H] |
| 0A | 11 | |
| 0B | BA | ADD IX,01H |
| 0C | 01 | |
| 0D | 30 | BA [07H] |
| 0E | 07 | |
| 0F | BA | ADD IX,FFH |
| 10 | FF | |
| 11 | 0F | HLT |

(b) 表 7 の 02,03 番地では IX に (01H) 番地の値を足し算している。つまり、IX に (01H) 番地の値を入れている（図 1 の 3）

(c) 表 7 の 04,05 番地では IX がゼロフラグ立っていたなら、PC の [0F] に移動する。IX には割る数があるので、値が 00H かどうかの判定を行っている（図 1 の 4）

(d) 表 7 の 06 番地では IX の値の初期化を行っている。IX には商が入るため、初期化する必要がある。（図 1 の 5）

(e) 表 7 の 07,08 番地では ACC の値の引き算を行っている。（図 1 の 6）

(f) 表 7 の 09,0A 番地ではネガティブフラグが立っていたなら、プログラムの終了となる。ACC の値がマイナスになる時はこれ以上引くことが出来ないことを示している（図 1 の 7）

(g) 表 7 の 0B,0C 番地は IX に 01H を足している。IX に値が入ることで商を求めることが出来る（図 1 の 8）

(h) 表 7 の 0D,0E 番地は PC の [07H] に JUMP する（図 1 の 8→6）

(i) 表 7 の 0F,10 は IX に FFH を足している。これは IX の値が 00H の時に実行される。（図 1 の 9）

5. (a)〜(e) のフローチャートの説明 (図 1 での説明)

(a) m > n で n が m を割り切れる場合の動作

i. m=14H(20),n=05H(5) の時の答えが 04H(4)

ii. 1→2→3→4→5　の順番で進む

iii. 6→7→8　を 4 回ループする (IX = 04H となる)

iv. 5 週目で ACC の値マイナスとなり　6→7→10　で終了する。

(b) m > n で n が m を割り切れない場合の動作

i. m=15H(21),n=05H(4) の時の答えが 05H(5)

ii. 1→2→3→4→5　の順番で進む

iii. 6→7→8　を 5 回ループする (IX = 05H となる)

iv. 6 週目で ACC の値マイナスとなり　6→7→10　で終了する。

(c) m < n の場合の動作

i. m=00H(0),n=01H(1) の時の答えが 00H(0)

ii. 1→2→3→4→5　の順番で進む

iii. 6 を通ると ACC の値がマイナスとなり　6→7→10　で終了する

(d) m=n の場合の動作

i. m=01H(1),n=01H(1) の時の答えが 01H(1)

ii. 1→2→3→4→5　の順番で進む

iii. 6→7→8→6→7→10　で進む

iv. 値が同じなので 8 は一度しか通らない

(e) n = 0 の場合の動作

i. m=01H(1),n=00H(0) の時の答えが FFH(255)

ii. 1→2→3 と進む

iii. 4 の時にゼロフラグが立っているので 9 に移動する

iv. 4→9→10　で終了となる

# 4　考察

## 4.1　実験 (1),(2),(3) の考察について

P0,P1,P2,P3,P4 の各実行フェーズにおいて、どのような処理が行われているか、実行した命令の種類毎に説明せよ。
(ここでの表 3.1 , 表 3.2 , 表 3.3 , 表 3.4 . 表 3.5 , 表 3.6 , 表 3.7 は提出した表のことである)

1. SUB 命令フェーズのフェーズ表 (表 3.1)
   (a) P0：PC が 01 の状態である。ACC の値は 07H である。
   (b) P1：PC が 02H にインクリメントし、MAR は P0 の PC から値を受け取った
   (c) P2：MAR から読み取り IR に A2 の命令が読み込まれた
   (d) P3：PC が 03H にインクリメントし、MAR は P2 の PC から値を受け取った
   (e) P0：SUB が実行され、ACC の値が 02H となった
2. LD 命令 (即値アドレスモード) の実行フェーズ表 (表 3.2)
   (a) P0：PC が 01 の状態である
   (b) P1：PC が 02H にインクリメントし、MAR は P0 の PC から値を受け取った
   (c) P2：MAR から読み取り IR に 62 の命令が読み込まれた
   (d) P3：PC が 03H にインクリメントし、MAR は P2 の PC から値を受け取った
   (e) P0：LD が実行され、ACC の値が 05H となった
3. LD 命令 (絶対アドレスモード) の実行フェーズ表 (表 3.3)
   (a) P0：PC が 01 の状態である
   (b) P1：PC が 02H にインクリメントし、MAR は P0 の PC から値を受け取った
   (c) P2：MAR から読み取り IR に 64 の命令が読み込まれた
   (d) P3：PC が 03H にインクリメントし、MAR は P2 の PC から値を受け取った
   (e) P4：MAR に LD 先の [07H] を格納している
   (f) P0：LD が実行され、ACC の値が FFH となった ([07H] には FFH が格納されていた)
4. SCF 命令実行フェーズ表 (表 3.4)
   (a) P0：PC が 01 の状態である
   (b) P1：PC が 02H にインクリメントし、MAR は P0 の PC から値を受け取った
   (c) P2：MAR から読み取り IR に 2F の命令が読み込まれた
   (d) P0：FLAG が 08(1000(2 進数)) となり、SCF によりの CF が立ったことがわかる
5. AND 命令実行フェーズ表 (表 3.5)
   (a) P0：PC が 01 の状態である
   (b) P1：PC が 02H にインクリメントし、MAR は P0 の PC から値を受け取った
   (c) P2：MAR から読み取り IR に E2 の命令が読み込まれた
   (d) P3：PC が 03H にインクリメントし、MAR は P2 の PC から値を受け取った

(e) P0：FLAG が 01(0001(2 進数)) となり、ZF が立ったことがわかる。

(f) ACC は 00H だったので AND で値が 00H となった

6. BZ 命令 (分岐条件不成立時) の実行フェーズ表 (表 3.6)

(a) P0：PC が 00 の状態である。IX の値は 02H である。

(b) P1：PC が 01H にインクリメントし、MAR は P0 の PC から値を受け取った

(c) P2：MAR から読み取り IR に AA の命令が読み込まれた

(d) P3：PC が 02H にインクリメントし、MAR は P1 の PC から値を受け取った

(e) P0：SUB 命令で IX の値が 01H となる

(f) FLAG が立っていないので、BZ はスルーされると考えられる

7. BZ 命令 (分岐条件成立時) の実行フェーズ表 (表 3.6)

(a) P0：PC が 00 の状態である。IX の値は 02H である。

(b) P1：PC が 01H にインクリメントし、MAR は P0 の PC から値を受け取った

(c) P2：MAR から読み取り IR に AA の命令が読み込まれた

(d) P3：PC が 02H にインクリメントし、MAR は P1 の PC から値を受け取った

(e) P0：SUB 命令で IX の値が 01H となる

(f) FLAG が立っていないので、BZ で JUMP すると考えられる

## 4.2 実験 (4) の考察について

各自が作成したアセンブラプログラムについて、可読性や実行効率を改善するための工夫ができないか考察せよ。

1. 単純に可読性、実行効率を良くするためにはソースを短くする必要がある。
2. 作成したプログラムは最短であると考えてる。
3. これ以上短くすることができないので、可読性は高いと考えられる。
4. 実行効率を良くするためにはループするときのフェーズ数が短い方が良い。
5. BN を使うのではなく、ZP を使って引き算をした直後でループを行うことでフェーズ数が少なくなると考えられる。(図 2)
6. ただし、これはループ内だけなので、ループの回数が少ないと表 7 のプログラムの方が効率がよい。

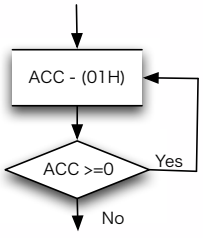


図 2: ループ中のプログラム

### 4.3　その他の考察について

割り算プログラムで使った命令の変わりになる命令を考える

1. 今回使った命令は ADD、BZ、EOR、SUB、BN、BA である
2. 表 7 の 00 番地の ADD は LD に変更することが可能である。LD にすることにより、初期値を考えなくて良い。

3. 02 番地の ADD は LD に変更することはできない。LD に変更すると、ゼロフラグが立たなくなる。
4. 02 番地の ADD の変わりに CMP を使うことが可能である。CMP を使い比較を行うことよりゼロフラグを判断出来る
5. 06 番地の EOR の変わりに ADD などを使って「0」を入れることは出来るが、プログラムカウンタが 2 行になり、効率が悪くなる
6. 以上の考察を表 8 にまとめた

表 8: 変更可能のまとめ

| 番地 | 命令 | 変更可能 |
|---|---|---|
| 00 | ADD | LD |
| 02 | ADD | CMP |
| 06 | EOR | ADD |

# 5 調査課題

## 5.1 (a) プロセッサ (CPU) の性能を表す指標に関して、以下の設問に答えよ

1. プロセッサ (CPU) の性能を表す指標の一つに IPC(instructions per (clock) cycle) と呼ばれるものがある。この IPC とはどのような指標か調査し、説明せよ。また、プロセッサ (CPU) の性能を表す IPC 以外の指標を５つ以上挙げ、それぞれについて説明せよ。
   【注意】プロセッサ (CPU) の性能を表す指標なので、メモリ容量やハードディスク容量などは該当しない。

   (a) IPC について
      i. 動作中のプログラム間でデータをやり取りすること
      ii. 1 クロックあたりに実行可能な命令数
      iii. 値が大きいほど、短いサイクルで多くの命令を処理することができる

   (b) ５つの指標
      i. MIPS
         A. コンピュータが 1 秒間に何百万回命令を実行できるかを示す単位
         B. 値が大きいほど、処理速度は速くなる
      ii. FLOPS
         A. CPU が１秒間に処理できる浮動小数点演算の回数のこと
         B. 主にスーパーコンピュータの性能評価に使用される
      iii. パス
         A. コンピュータ内部でデータをやり取りするための信号の通路である
         B. 値が大きいほど一度に処理できるデータ量が増える
      iv. クロック周波数
         A. コンピュータの動作のタイミングを取るための周期的な信号の周波数
         B. 値が大きいほど処理速度は早くなる
      v. CPU の平均命令実行時間
         A. コンピュータが１命令するのにかかる時間
         B. 平均命令実行時間は MIPS 値の逆数である

2. IPC が１の CPU を載せたコンピュータ A と IPC が２の CPU を載せたコンピュータ B があり、両方のコンピュータで同じプログラムを同時に実行した。その結果、コンピュータ B の方が IPC が大きいにも関わらず、コンピュータ A の方が先に処理を終了した。この原因として考えられる状況や環境を３つ以上挙げて説明せよ。

   (a) MIPS の違い。コンピュータメーカーごとに命令体系が異なる。MIPS が 2 倍以上違えば、コンピュータ B の処理速度はコンピュータ A よりも早くなる

   (b) パスの違い。パスが違うことで、処理速度も変化する。バスの大きさが 2 倍以上大きいならば、送れる情報も 2 倍以上になる。

   (c) クロック周波数の違い。クロック周波数が 2 倍以上変わることにより、クロックの間隔が 2 倍以上短くなる。そうなることで、コンピュータ B の方が早くなるときがある。

## 5.2 前回の報告書で調査したパイプライン・アーキテクチャでは、処理の乱れを生じる危険性がある。このような、パイプライン処理における乱れの原因をパイプライン・ハザードという。パイプライン・ハザードの種類およびそれらの解消方法について調査し、図表等を用いてわかりやすく説明せよ。

### 5.2.1 ハザードの種類

1. 構造ハザード

   (a) コンピュータの内部構成が原因のハザードのことである。

   (b) あるステージを実行中の命令 A と別のステージを実行中の命令 B が同じハードウェア資源を使わなければならない場合などに生じる。

   (c) 図 3 に例を示した。

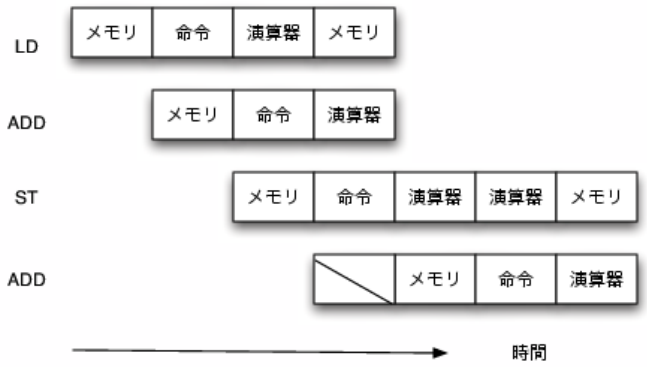


図 3: 構造ハザード

   (d) 図 3 の 4 行目の ADD で 1 行目と同じメモリの命令を行おうとしているため、構造ハザードが生じている

2. データハザード

   (a) 命令 A の実行結果を見なければ後続の命令 B が実行出来ない場合がある

   (b) 命令 A と命令 B の間に依存関係がある

   (c) データ依存のある 2 命令が接近している場合である

   (d) 図 4 に例を示した

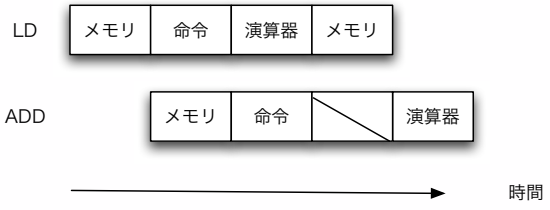


図 4: データハザード

(e) 図 4 の LD と ADD は依存関係である。

(f) LD のメモリがわからないと ADD の演算器が実行出来ない

3. 制御ザード

(a) 分岐命令は後続の命令がどれになるのかを決める。

(b) 分岐命令とのそれ以降の命令には依存関係がある

(c) 図 5 に例を示した

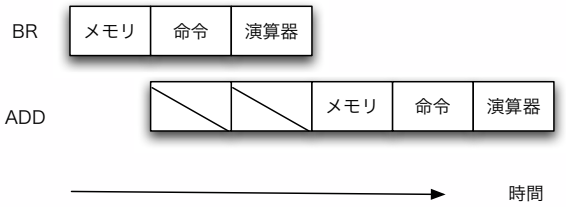


図 5: 制御ハザード

(d) 図 5 では BR を使って分岐をしている

(e) BR 命令が終了しない限り次の命令が実行されない

### 5.2.2 解消法

1. 資源を増やす (構造ハザード)

(a) 競合しないように資源を増やせば、同じ命令にあたることはない

(b) 図 6 に例を示した

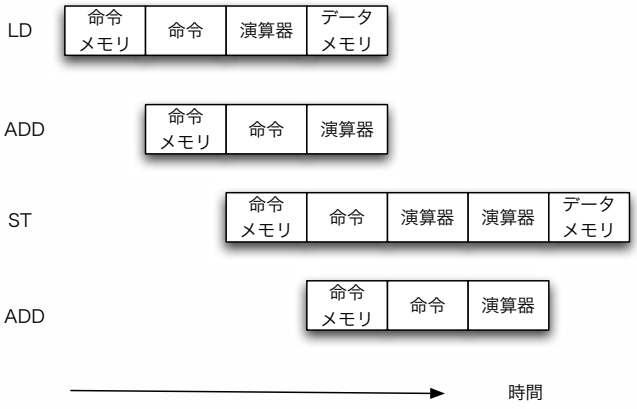


図 6: 資源を増やす

(c) 図 6 では「メモリ」を「命令メモリ」と「データメモリ」に分けた

(d) 「命令メモリ」と「データメモリ」は違うので構造ハザードは起きない

2. フォワーディング (データハザード)

(a) データハザードを検知したら実行結果を次の命令に直接渡す

(b) 演算装置によってハードウェア的に実現される

(c) 図 7 に例を示した

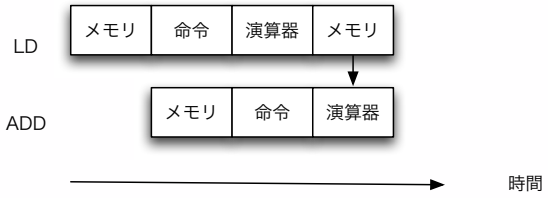


図 7: フォワーディング

(d) 書き込み前でも、直接結果を渡している

3. 分岐予測

(a) 分岐かどうかを予測して処理を進め、予測がはずれた場合に分岐命令以下の命令を破棄する

(b) 図 8、9 に例を示した

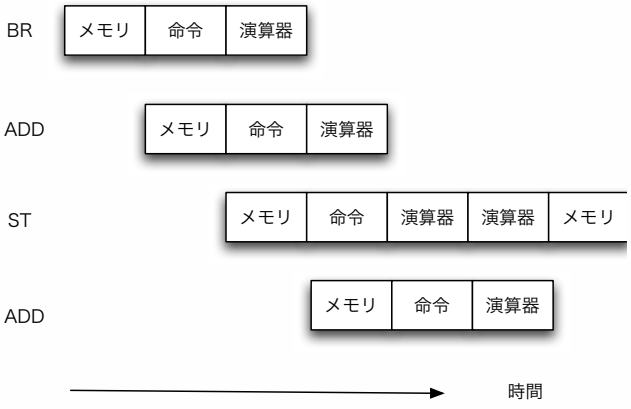


図 8: 分岐予測 (分岐しない場合)

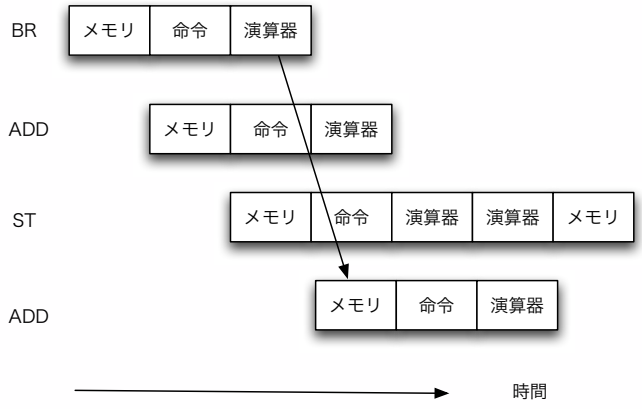


図 9: 分岐予測 (分岐する場合)

(c) 分岐を行わない場合 (図 9) はいつも通りに処理が流れる

(d) 分岐を行う場合 (図 8 では 1 から 4 行目に BR) は 2,3 行目は破棄される

# 6 感想

1. 実験について
   実験はとても楽しかったです。実行フェーズ表を埋める作業はめんどうでした。実行フェーズ表を埋めるのに相当時間がかかりました。いろいろとおかしなことが起きて大変でしたけど、いろいろと面白いことに気づけたので良かった思います。割り算のプログラムはとても簡単でした。パッと思いついたやつで実行したら出来たので、もう少し難しいプログラムを作成したかったです。でもなかなか面白い実験だったと思います。機械語に興味も持ちましたし、いろいろと理解しました。とても良かったと思います。

2. 報告書について
   報告書についてはめんどうな感じがしました。実行結果を書くのがとても大変でした。考察ではいろいろと新しい発見が出来たので面白かったです。調査課題はいつも通り調べるのがかなり面倒でした。書き終わってみたら、いろいろと学べて良かったのではないかと思います。吉田先生のレポートは書くのが面倒です。書いている間はものすごくやる気がなくなります。そこが大きな難点です。書き終わったときに得るものが多いのでそれはそれでいいと思います。

# 参考文献


wikipedia
http://ja.wikipedia.org/wiki/

CPU の性能
http://akademeia.info/index.php?CPU%A4%CE%C0%AD%C7%BD

コンピュータアーキテクチャの話 (124)
http://journal.mycom.co.jp/column/architecture/124/index.html

パイプラインハザード
http://akademeia.info/index.php?%A5%D1%A5%A4%A5%D7%A5%E9%A5%A4%A5%F3%A5%CF%A5%B6%A1%BC%A5%C9

コンピュータアーキテクチャ
編集：電子情報通信学会
著者：坂井修一