知能情報実験II　:　第三回レポート

175751C 宮城孝明

平成 30 年 10 月 26 日

目 次

1　実験目的

機械語 (マシン語) 命令をフェーズごとに実行させ, そのときのコンピュータ内部の状態を観測することにより, 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目的とする.

2　概要

今回の実験は,KUE-CJIP2 を用いて,1 クロックフェーズだけ命令実行し, プログラムカウンタやフラグレジスタ,ACC,IX,MAR,IR の観測した. 毎回を手入力し, 計 7 回行った. 次の実験としては, ソートプログラムの作成を行った. データ領域に 3 個のデータを予め格納しており, プログラムの実行した時に, データを取り出しソートを行うプログラムであった. 比較を行い交換しないといけないため, 比較の処理を行い次の処理でフラグレジスタの符号がプラスの時の処理, マイナスの時の処理をぞれそれ考える必要がある. 最後の実験は,2 つの非負整の商を求めるプログラムを作成するものであった.2 つの整数関係だけでも 5 つのパターンがあるため, 処理が 5 パターンある. 商を求めるため, 割られる数を割る数から引き, 割られる数がマイナスになるまでこれを繰り返す. これにより, 処理を完成させた.

3　実験結果

1. 実験 (1),(2),(3) の結果について

   提出しているため, 報告は割愛する.

2. 実験 (4) について

   降順のプログラム

   図 2 のフローチャートは, 降順プログラムの実行を表している. まず初めの命令は, データ領域 (00H) 番地の値を ACC に,(01H) 番地の値を IX に格納する. そして, 格納した ACC と IX を比べ,ACC が小さければ,ACC の値を (01H) 番地,IX の値を (00H) 番地に格納し, それぞれを入れ替えする.ACC が大きければ, 降順になっているためそのままで良いので, 特に何もしないで次の命令に移る. 次の命令では, データ領域 (01H) 番地の値を ACC に,(02H) 番地の値を IX に格納する. そして, 先ほどと同様に ACC と IX を比べ,ACC が小さければ,ACC の値を (02H) 番地,IX の値を (01H) 番地と入れ替えを行う. 入れ替えが終了したら, 一番最初の処理に戻る. 二回目の比較で

ACC が大きければ, 降順に並んでいることになるため処理は全て終了する.

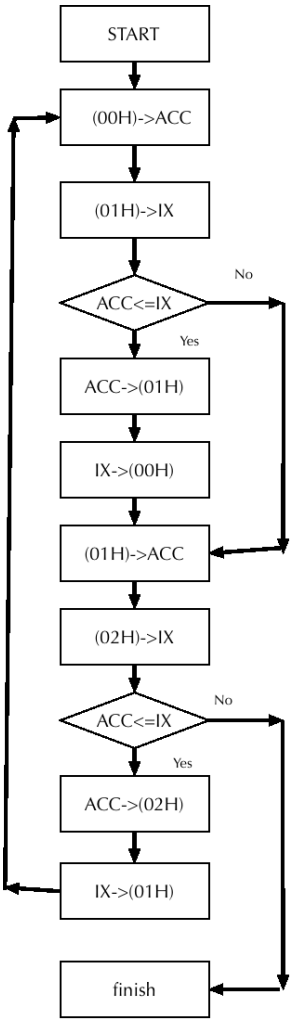

図 1: 実験 (4) のフローチャート

Listing 1: 実験 (4) のコード

```
ADRS DATA OPECODE
  00  65  LD ACC,(00H)
  01  00
  02  6D  LD IX,(01H)
  03  01
  04  F1  CMP ACC,IX
  05  33  BP
  06  0B
  07  75  ST  ACC,(01H)
  08  01
  09  7D  ST IX,(00H)
  0A  00
  0B  65  LD ACC,(01H)
  0C  01
  0D  6D  LD IX,(02H)
  0E  02
  0F  F1  CMP ACC,IX
  10  33  BP
  11  18
  12  75  ST ACC,(02H)
  13  02
  14  7D  ST IX,(01H)
  15  01
  16  30  BA
  17  00
  18  0F  HLT
```

3. 実験 (5) について

   データ領域 (00H) に m の値,(01H) に n の値を格納する.

   (a) の場合の処理の手順

   m に 14H(20),n に 05H(5) の値を格納する.始めに IX と IX を XOR して値を 0 にする. そして,ACC にデータ領域 (01H) の値を代入る.ACC の値は 0 ではないため, そのまま処理を下に降りる. その次に,ACC にデータ領域 (00H) の値を代入する. そして, 現在の ACC の値をデータ領域 (01H) に格納されている値から引く. そして, その値を ACC に格納する.ACC が 0 未満ではないため, 処理は下の階層に続く. そして,IX(値は 0) に+1 ずつ加算していく. この処理は,ACC の値がマイナスになるまで続ける. マイナスになったとき,IX の値には m と n の割り算の商の値が格納されている. これにより,IX の値を (02H) に格納し, 処理を終了する. 結果は,4 である.

(b) の場合の処理の手順

m が 15H(21) で,n が 04H(4) の値とする. 最初の命令で,IX の中身を 0 にする. そして, データ領域 (01H) の値を格納する.(01H) は,0 出ないためそのまま降りていく. そして,ACC にデータ領域 (00H) の値を格納する. そして,ACC 値から (01H) を引く. もし,ACC の値がマイナスになるならジャンプ命令をするが,ACC の値はプラスのままであるため,IX に 1 加える命令を実行する. それを ACC がマイナスになるまで続ける.IX の値が 5 になる時この処理は終了する. そんため, ジャンプ命令を実行し,IX をデータ領域 (02H) に格納し終了する. 結果は,5 である.

(c) の場合の処理の手順

m に 00H(0),n に 01H(1) を格納している. 最初の命令で,IX の中身を 0 にする. そして, データ領域 (01H) の値を格納する.(01H) は,0 出ないためそのまま降りていく. そして,ACC にデータ領域 (00H) の値を格納する. そして,ACC 値から (01H) を引く. マイナスになってしまうのですぐにジャンプ命令に移る. この時,IX には何も加えられていながデータ領域 (02H) に値を格納する. そして, 処理を終了する. 結果は,0 である.

(d) の場合の処理の手順について

m に 01H(1),n に 01H(1) を格納する. 最初の命令で,IX の中身を 0 にする. そして, データ領域 (01H) の値を格納する.(01H) は,0 出ないためそのまま降りていく. そして,ACC にデータ領域 (00H) の値を格納する. そして,ACC 値から (01H) を引く. マイナスにならないため IX に 1 加える. 次の命令でマイナスになり, ジャンプ命令を実行し,IX の値をデータ領域に格納し, 終了する. 結果は,1 である.

(e) の場合の処理の手順

m に 01H(1),n に 00H(0) を格納する. 最初の命令で,IX の中身を 0 にする. そして, データ領域 (01H) の値を格納する.ACC の値は 0 のためジャンプ命令を実行する. そして,IX に FFH 代入し,IX の値をデータ領域 (02H) に格納し終了する.

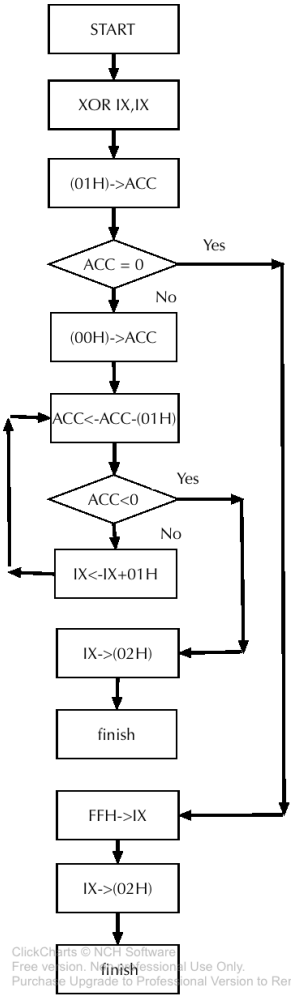

図 2: 実験 (5) のフローチャート

Listing 2: 実験 (5) のコード

```
 1  ADRS DATA OPECODE
 2    00   C9  XOR IX,IX
 3    01   65  LD ACC,(01H)
 4    02   01
 5    03   F2  CMP ACC,00H
 6    04   00
 7    05   39  BZ
 8    06   14
 9    07   65  LD  ACC,(00H)
10    08   00
11    09   A5  SUB ACC,(01H)
12    0A   01
13    0B   3A  BN
14    0C   11
```

```
15    0D  BA  ADD IX,01H
16    0E  01
17    0F  30  BA
18    10  09
19    11  7D  ST  IX,(02H)
20    12  02
21    13  0F  HLT
22    14  6A  LD  IX,FFH
23    15  FF
24    16  7D  ST  IX,(02H)
25    17  02
26    18  0F  HLT
27
28  -----------------
29    00  14
30    01  05
31  -----------------
32    00  15
33    01  04
34  -----------------
35    00  01
36    01  01
37  -----------------
38    00  01
39    01  01
40  -----------------
41    00  01
42    01  00
```

4 考察

1. 実験 (1),(2),(3) の考察について

   命令には,4 つの段階がある. 命令の読み出し, 命令の解読, 命令の実行, 実行結果の格納に分けられる.P0 の実行フェーズは, 命令の読み出しが行われている. そのため,PC の値が MAR に格納される. 次の P1 の実行フェーズは命令の解読のため,MAR の値から持ってこられた命令コードを IR(インストラクションレジスタ) に格納されるその次の P2 フェーズでは. 命令の実行を行う. 解読された命令で何を行うのかを決め, 値を取り出してきて, 命令実行を行う. 最後の P3 フェーズでは, 命令の格納を行う. 先ほどの命令実行で出た結果をレジスタに格納し, 処理を終了する. しかし,(2) の 2 の命令は, データ領域から値を取り出すという作業が含まれるため他の処理も 1 つフェーズが多い. 同じ (2) の 3 命令は, セットする値を変更するだけの処理のため, 他の命令よりも 1 フェーズ少ない.

   PC は, プログラムカウンタと呼ばれている. 役割としては, 次の実行

する命令のメモリ番地を格納するレジスタである.

FLAG は, フラグと呼ばれている.2 つの値か状態を 0 か 1 で判断する. 命令分岐などで使われる.

MAR は, メモリアドレスレジスタと呼ばれている. 実行するのに必要な値が格納されているメモリ番地を格納するレジスタである.

IR は, インストラクションレジスタと呼ばれている. 命令コードを格納するレジスタである.

2. 実験 (4) の考察について

   3 個のデータの並べ替えを限定すれば, このプログラムの修正する必要ないと判断する. このプログラムは, バブルソートを使用しているため, 始めから揃っている場合でも最初から最後まで処理を通してしまうため時間を掛けてしまう. もし, 始め揃っていたら処理をしないで終了のところまで移動するプログラムを作ろうとしたら, 今よりもコード数が増えてしまい, 可読性を損ねてしまう.

   3 個のデータの状態だと比較の命令が 2 つで十分だが,4 個になると比較は 3 つ以上になる.5 個だと 4 つは必要になる. しかし, 上から下の処理を繰り返す条件を立てるのは厳しいため, 上から下から並べ替えを行い, 下から上に並べ替えを行う. こうすることで, 降順に並べ替えが完成する. しかし, 要素が増えれば増えるほどコード数がとても長くなるので良いコードとは思わない.

3. 実験 (5) の考察について

   改善する点としては, 終了コードが 2 つあるところを 1 つにする必要ことが改善点である.ACC が 0 の時すぐに IX に FF を格納し, ジャンプ命令を行い終了する.

4. その他の考察について

   本実験を通し, ソート処理のプログラミングを作成する際に, 要素が増えた場合を考えたプログラムを作っていかないといけない. 決まった要素だけで考えるプログラムだと, 要素の増減際に対応できない. 枠にはまった考えだと応用が利かないと知った.

5 調査課題

1. パイプラインハザードについての対策

   パイプラインハザードは, 次の 3 種類に分けられる. 構造ハザードとデータハザード, 制御ハザードである.

構造ハザードは, コンピュータの構造自体に問題がある. 例としては, メモリの格納と読み込みが同時には実行できないという問題点である. これを解決するためには, 資源の多重化をする. つまり, データメモリと命令メモリに分け, アクセスするための制御線を整備することで解決することができる.

データハザードは, 命令 A の実行結果がないと命令 B を実行することができないということで発生する問題である. つまり, 命令 A と命令 B が依存関係にあると言える. この問題を解決するために, フォワーリングという方法がある. これは,E ステージの結果を W ステージを通すことせず, 次の命令の E ステージに送る. 処理を行い, 結果が前の命令と同じであれば, 結果として出力する.

制御ハザードは, 分岐命令の際に後続のどの命令になるかを決める。この時にも依存関係が発生する. 無条件分岐の時には, 命令アドレス生成するタイミングがわかる.D ステージの終わりで行うことができるため, 効率化することができる. 条件分岐の時は, 遅延分岐と分岐予測を使用する. 遅延分岐は, 分岐するしないに関わらず, 分岐命令直後に共通命令を実行しておくという方法である. 分岐予測は, 分岐が起きるか起きないかを予測し, 外れた場合は分岐命令以下の命令を消すという方法である. 予測の仕方は, 分岐するしないかのどちらか一方に固定するものと, 過去の履歴から決めるものなどさまざまである. 最後にソフトウェアを用いて, 依存関係の命令同士をなるべく離し, ハザードを起きにくくする命令スケジューリングという方法もある. この方法は, コンパイラが行う.

2. ICP

   コンピュータ B に構造ハザードが起きて, 処理に時間がかかった.

   コンピュータ A は, 並列処理を行いコンピュータ B よりも速く実行が行えた.

   コンピュータ B が, 空き容量をうまく活用できなかった.

6 感想

今回の実験では, 先のことを考えたプログラムをかけるようにならないと, 後からのプログラムの拡張の際にとても苦労すると感じた.

7　参考文献

参考文献

[1] wiki; https://ja.wikipedia.org/wiki/命令レジスタ

[2] tosiba; https://toshiba.semiconstorage.com/jp/designsupport/e-learning/micro_intro/chap4/1274772.html

[3] tokyo; https://www.mtl.t.utokyo.ac.jp/ sakai/hard/hard4.pdf

[4] コンピュータアーキテクチャ; 坂井 修一 2018 年 1 月 15 日 コンピュータアーキテクチャ