情報工学実験2

ーコンピュータアーキテクチャ編ー
命令実行フェーズ

075730G:澤岨千明

実験日 2008/10/27
締切日 2008/11/17
提出日 2008/11/28

共同実験者
075711A　山口　藍
075715C　嘉陽　裕香
075722F　斉藤由利絵

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、機械声命令の実行の仕組みを理解することを目的とする。

実験概要

KUE-CHIP2 を用いて実験を行う。予め示されているアセンブリプログラムをハンドアセンブルで機械語にしたあと実際に動かす。ここで注意するのは、プログラムを 1 フェーズずつ実行させるということである。1 フェーズずつ実行し、その際の PC、FLAG、ACC、IX、MAR、IR の状態を確認していく。

複数のプログラムを実行、各フェーズでの処理を観測した後、自分で商 m ÷ n を求めるアセンブリプログラムを作成し、動作を確認する。

実験内容

(1) 以下に示すプログラムを用いて、減算命令 (SUB) の実行フェーズを観測せよ。ただし、ACC には予め 07H を格納しておき、(7 − 5) の計算過程を観測するものとする。

番地	機械語	アセンブリ言語	
00	00	NOP	＊3 フェーズ
01	A2	SUB ACC, 05H	＊4 フェーズ
02	05		
03	0F	HLT	＊3 フェーズ

(2) 下記の 4 つのプログラムについて、それぞれ 2 番目の命令の実行フェーズを観測し、実行フェーズ表を完成させよ。また、それぞれのアセンブラプログラムに対応する機械語プログラムをしめせ。

```
1.  NOP
    LD ACC, 05H
    HLT
```

```
2.  NOP
    LD ACC, [07H]
    HLT
```

```
3.  NOP
    SCP
    HLT
```

```
4.  NOP
    AND ACC, 05H
    HLT
```

(3) 下記のプログラムにおいて、IX = 02H とした場合および IX = 01 とした場合のそれぞれについて、BZ 命令の実行フェーズを観測し、実行フェーズ表を完成させよ。なお下記のプログラムにおいて、jp はラベルであり、BZ 命令の分岐先のアドレスを表すものとする。機械語プログラム作成時に、各自で適当な値を jp に設定すること。

```
    SUB IX, 01H
    BZ  jp
    HLT
jp: HLT
```

(4) 8 ビットの 2 進数 m(データ領域の 00H 番地に格納)、n(データ領域の 01H 番地に格納) に対し、商 m ÷ n(小数点以下は不要) を求めるアセンブラプログラムを作成し、下記の場合の動作を確認しなさい。ただし、n = 0 のときの商を FFH とすること。

(a) m ＞ n で、n が m を割り切れる場合の動作

(b) m ＞ n で、n が m を割り切れない場合の動作

(c) m ＜ n の場合の動作

(d) m ＝ n の場合の動作

(e) n ＝ 0 の場合の動作

実験結果

実施した実験項目 (実験 (1)〜(4)) を全て示し、以下の要領で各結果について報告せよ。

- 実験 (1) の結果について

◆ SUB 命令

番地	機械語	アセンブリ言語
00	00	NOP
01	A2	SUB ACC 05H
02	05	
03	0F	HLT

表 3.1：SUB 命令の実行フェーズ表

フェーズ	LED	PC	FLAG	ACC	IX	MAR	IR
実行直前	P0 点灯	01	00	07	00	00	00
P0 実行後	P1 点灯	02	00	07	00	01	A2
P1 実行後	P2 点灯	02	00	07	00	01	A2
P2 実行後	P3 点灯	03	00	07	00	02	A2
P3 実行後	P4 点灯						
P4 実行後	P0 点灯	03	00	02	00	02	A2

- 実験 (2) の結果について

◆ LD 命令 (即値アドレスモード)

番地	機械語	アセンブリ言語
00	00	NOP
01	62	LD ACC, 05H
02	05	
03	0F	HLT

表 3.2：LD 命令 (即値アドレスモード) の実行フェーズ表

フェーズ	LED	PC	FLAG	ACC	IX	MAR	IR
実行直前	P0 点灯	01	00	00	00	00	00
P0 実行後	P1 点灯	02	00	00	00	01	00
P1 実行後	P2 点灯	02	00	00	00	02	62
P2 実行後	P3 点灯	03	00	00	00	02	62
P3 実行後	P4 点灯						
P4 実行後	P0 点灯	03	00	05	00	02	62

◆ LD 命令 (絶対アドレスモード)

番地	機械語	アセンブリ言語
00	00	NOP
01	64	LD ACC, [07H]
02	07	
03	0F	HLT

表 3.3 : LD 命令 (絶対アドレスモード) の実行フェーズ表

フェーズ	LED	PC	FLAG	ACC	IX	MAR	IR
実行直前	P0 点灯	01	00	00	00	00	00
P0 実行後	P1 点灯	02	00	00	00	01	00
P1 実行後	P2 点灯	02	00	00	00	01	64
P2 実行後	P3 点灯	03	00	00	00	02	64
P3 実行後	P4 点灯	03	00	00	00	07	64
P4 実行後	P0 点灯	03	00	FF	00	07	64

◆ SCF 命令

番地	機械語	アセンブリ言語
00	00	NOP
01	2F	SCF
02	0F	HLT

表 3.4 : SCF 命令の実行フェーズ表

フェーズ	LED	PC	FLAG	ACC	IX	MAR	IR
実行直前	P0 点灯	01	00	00	00	00	00
P0 実行後	P1 点灯	02	00	00	00	01	00
P1 実行後	P2 点灯	02	00	00	00	01	2F
P2 実行後	P3 点灯						
P3 実行後	P4 点灯						
P4 実行後	P0 点灯	02	08	00	00	01	2F

◆ AND 命令

番地	機械語	アセンブリ言語
00	00	NOP
01	E2	AND ACC, 05H
02	05	
03	0F	HLT

表 3.5：AND 命令の実行フェーズ表

フェーズ	LED	PC	FLAG	ACC	IX	MAR	IR
実行直前	P0 点灯	01	00	00	00	00	00
P0 実行後	P1 点灯	02	00	00	00	01	00
P1 実行後	P2 点灯	02	00	00	00	01	E2
P2 実行後	P3 点灯	03	00	00	00	02	E2
P3 実行後	P4 点灯						
P4 実行後	P0 点灯	03	01	00	00	02	E2

- 実験 (3) の結果について

◆ BZ 命令

番地	機械語	アセンブリ言語	
00	AA		SUB IX, 01H
01	01		
02	39		BZ jp
03	0		
04	0F		HLT
05	0F	jp：	HLT

表 3.6：BZ 命令 (分岐条件不成立時) の実行フェーズ表

フェーズ	LED	PC	FLAG	ACC	IX	MAR	IR
実行直前	P0 点灯	00	00	00	00	00	00
P0 実行後	P1 点灯	01	00	00	00	00	00
P1 実行後	P2 点灯	01	00	00	00	00	8A
P2 実行後	P3 点灯	02	00	00	00	01	8A
P3 実行後	P4 点灯						
P4 実行後	P0 点灯	02	0A	00	FF	01	8A

表 3.7：BZ 命令 (分岐条件成立時) の実行フェーズ表

フェーズ	LED	PC	FLAG	ACC	IX	MAR	IR
実行直前	P0 点灯	02	02	02	FF	01	AA
P0 実行後	P1 点灯	03	02	02	FF	02	AA
P1 実行後	P2 点灯	03	02	02	FF	02	39
P2 実行後	P3 点灯	04	02	02	FF	03	39
P3 実行後	P4 点灯						
P4 実行後	P0 点灯	04	02	02	FF	03	39

- 実験 (4) の結果について

  プログラムおよびフローチャートを以下にしめす。

番地	機械語	アセンブラ言語	
00	C9		EOR IX, IX
01	65		LD ACC, (01H)
02	01		
03	A2		SUB ACC, 00H
04	00		
05	39		BZ jp1
06	18		
07	65		LD ACC, (00H)
08	00		
09	BA	jp3 :	ADD IX, 01H
0A	01		
0B	A5		SUB ACC, (01)
0C	01		
0D	3B		BZN jp2
0E	11		
0F	30		BA jp3
10	09		
11	C2	jp2 :	EOR ACC, 00H
12	00		
13	39		BZ jp4
14	1A		
15	AA		SUB IX, 01H
16	01		
17	0F		HLT
18	6A	jp1 :	LD IX, FFH
19	FF		
1A	0F	jp4 :	HLT

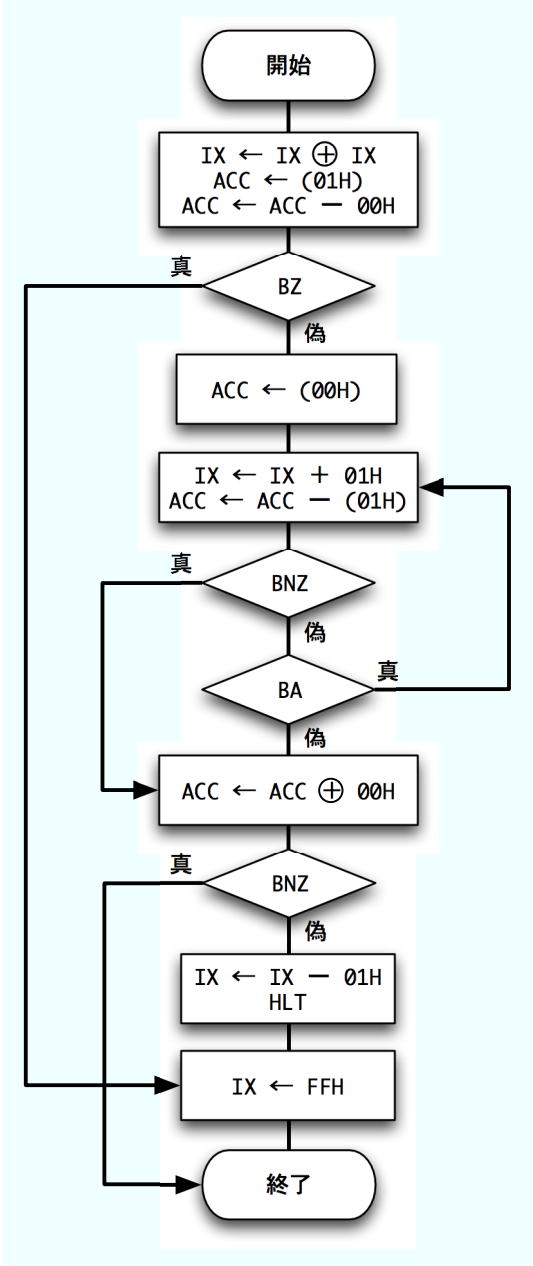

図 3.1：実験 4 のプログラムのフローチャート

m ÷ n という計算をするプログラムである。予めデータ領域の 00H 番地に値 m、01H 番地には値 n を格納しておく。

まず IX を初期化し ACC に 01H の値、つまり n を格納する。そして ACC から 00H を引き、0 かどうかを判断する。このとき n=0 ならば、IX に FFH を格納し演算は終了する。

そうでなければ、改めて ACC に 00H の値、つまり m を格納する。IX に 01H を加算していき、同時に ACC(m) から 01H の値 (n) を減算していく。このとき、ACC の値 (m) が 0 でないときは BNZ 命令、BA 命令によりループし IX と ACC の演算を繰り返す。この何回減算が行われたかを数える IX がこの演算の商である。

ACC の値が 0 であるとき、ACC を 00H で EOR をとる。この値が 0 であるならば、丁度割り切れた解であるのでそのまま終了する。0 でなければ、余分に減算したということになるので IX から 01H を引き、終了する。

考察

上記の実験結果に基づいて、以下の点について詳しく述べよ。

- 実験 (1)、(2)、(3) の考察について

PC プログラムカウンタ
次に取り出す命令のアドレスを記憶するレジスタ

FLAG フラグレジスタ
演算結果の状態によって、特定のビットが１や０になるレジスタ。

ACC アキュムレータ
データの演算を行う場合に演算装置によって利用されるレジスタ

IX インデックスレジスタ
主記憶上でのデータのアドレスを求めるときなどに、基準からの増減値を記憶するレジスタ

MAR メモリアドレスレジスタ
アクセスすべきアドレスを格納したあと実際にアクセスを行うレジスタ

IR 命令レジスタ
取り出した命令そのものを記憶するレジスタ

SUB 命令
P0：MAR に PC の値が格納され、PC の値がインクリメントされる。ACC には予め 07H が格納されている。
P1：IR に SUB 命令の機械語が格納されている。
P2：MAR に PC の値が格納され、PC の値がインクリメントされている。
P3：処理されいない。
P4：ACC に SUB 命令の演算結果が格納されている。

LD 命令 (即値アドレスモード)
P0：MAR に PC の値が格納され、PC の値がインクリメントされる。
P1：IR に LD 命令 (即値アドレスモード) の機械語が格納されている。
P2：PC がインクリメントされている。
P3：処理されいない。
P4：ACC に 05H が格納されている。

LD 命令 (命令アドレスモード)
P0：MAR に PC の値が格納され、PC の値がインクリメントされている。
P1：IR に LD 命令 (絶対アドレス) の機械語が格納されている。
P2：MAR に PC の値が格納され、PC の値がインクリメントされている。
P3：MAR に 07H というアドレスが格納されている。
P4：ACC に MAR に格納されているアドレス値 FFH が格納されている。

SCF 命令
P0：MAR に PC の値が格納され、PC の値がインクリメントされている。
P1：IR に SCF 命令の機械語が格納されている。
P2：処理されない。

P3：処理されない。
P4：FLAG に 08H が格納される。

AND 命令
P0：MAR に PC の値が格納され、PC の値がインクリメントされている。
P1：IR に AND 命令の機械語が格納される。
P2：MAR に PC の値が格納され、PC の値がインクリメントされている。
P3：処理されない。
P4：FLAG に 01H が格納される。

BZ 命令 (分岐条件不成立時)
予め、IX に 02H を格納しておく。
P0：PC がインクリメントされる。
P1：IR に SUB 命令の機械語が格納される。
P2：MAR に PC の値が格納され、PC の値がインクリメントされている。
P3：処理されない。
P4：FLAG に 0AH が格納され、IX に FFH が格納される。

BZ 命令 (分岐条件成立時)
予め、IX に 01H を格納しておく。
P0：MAR に PC の値が格納され、PC の値がインクリメントされている。FLAG には 02H、ACC に 02H、IX に FFH、が格納されており、IR には SUB 命令の機械語が格納されている。。
P1：MAR に PC の値が格納され、PC の値がインクリメントされている。IR に BZ 命令の機械語が格納される。
P2：MAR に PC の値が格納され、PC の値がインクリメントされている。
P3：処理されない。
P4：P2 と変化はみられない。

- 実験 (4) の考察について
  うまく条件を指定し使えばいくらか効率を改善できると思われる。
- その他の考察について
  - 本実験を通して得られた知見について詳しく説明せよ。
    命令の実行はいくつかのフェーズに分けられ、またどのように実行されていくのかを知る事ができた。仕様書の最終ページにあるフェーズ表をより理解することができた。

調査課題

下記の各項目について調査し、その結果を報告せよ。

(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) の性能を示す指標なので、メモリ容量やハードディスク容量などは該当しない。

IPC（ Instructions Per Cycle ）

CPUが処理できるクロックあたりの命令実行数を表す指標。

命令実行数 ÷ クロック数 でもとめることができる。IPCが高いほど、１クロックの間で実行できる命令が多くなる。

FLOPS（ Floating point number Operations Per Second ）

１秒間に浮動小数点数演算が何回できるかという能力を理論的/実際的に表した指標。１ FLOPSのコンピュータは一秒間に１回の浮動小数点数演算ができる。大型コンピュータの性能指標として用いられることが多い。

MIPS（ Million Instructions Per Second ）

１秒間に何百万個の命令が実行できるかを示す指標。

１ MIPSのコンピュータは１秒間に100万回の命令を処理できる。

iCOMP（ Intel COmparative Microprocesser Performance ）

Intel社の開発した同社のX86形マイクロプロセッサの性能指標。

整数演算、浮動小数点演算、グラフィックス処理などの性能を計測する。いくつかのバージョンがあり、それぞれの基準値を100とした場合の相対値で表される。

モデルナンバー（ model number ）

AMD社のマイクロプロセッサ「Athlon XP」で、同社が動作周波数の代わりに使用されている性能指標値。

このモデルナンバーとは、元々はメーカーが製品を区別するためにつける数列名で、数値自体には量的な意味はないものであった。

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。

- CPUの種類

  CPUの種類によってクロック周波数が異なるので、IPCが２であるコンピュータBのクロック周波数が遅い物であれば、IPCが１のコンピュータA方が速く命令処理を行う場合がある。

- FSB (Front Side Bus)

  FSBとはパソコン内部でメインメモリなどがCPUとデータをやり取りするバスの動作周波数のこと。このFBSが高けいほど速度は増すのでコンピュータAの方が高ければ、こちらの方が先に処理を終了する場合がある。

- ２次キャッシュ

  ２次キャッシュとは、マイクロプロセッサ内部に記憶装置である。これを用いることで、低速なメインメモリのアクセスを減らし処理速度を早めることができる。このキャッシュ容量が小さい場合、コンピュータBの処理が遅れる場合がある。

(b) 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法法について調査し、図表等を用いて分

かりやすく説明せよ。

パイプライン処理を行うと、一定数の命令がプロセッサ内で同時に動作できるようになるので、性能を向上させることができる。
だが、複数の命令同士が持つ依存状態から命令の投入を中断せざるを得ない状況が生じ、処理速度の低下に繋がる。これをパイプラインハザードと呼ぶ。

パイプラインハザードの種類

IF：命令を命令キャッシュから読み出す
ID：レジスタファイルをレジスタ指定子で参照
EX：数値の計算を行う
MA：ロード・ストアを行う
WB：レジスタにデータを書き込む

○ データハザード
処理するデータの依存関係に起因するハザード

	1	2	3	4	5	6	7	8	9
add r1, r2, r3	IF	ID	EX	MA	WB				
and r4, r1, r5		IF	ID	ID	ID	ID	EX	MA	WB

上の表から分かることだが二つ目の命令で用いられているレジスタ 1 は最初の命令でも用いられている。まだ最初の命令は処理されていないので、二つ目の命令を処理することができない。そのため、最初の命令が処理されるまでパイプラインを止める必要がでてくる。
レジスタが問題になる場合は WB → ID で、メモリが問題の場合は MA → MA という簡形で発生する。

解消法
プログラムをコンパイルする際などに、プログラムの内容に影響のない範囲で CPU に送る命令の順序を調整し、データハザードを起きにくいようにする。データを書き込むのと同時にパイプラインに投入する方法もある。

○ 構造ハザード
ハードウェア的な資源の競合に起因するハザード

	1	2	3	4	5	6	7	8
命令 1	IF	ID	EX	MA	**WB**			
命令 2		IF	ID	EX	MA	WB		
命令 3			IF	ID	EX	MA	WB	
命令 4				IF	**ID**	EX	MA	WB

上の表の時間 5 の列に注目する。この列で WB と ID が使われている。この二つは同一の部品 (ハードウェア資源) を要求するため、資源の競合が発生する。

解消法
一つの時間単位をさらに二つに分けて、その前半に WB、後半に ID を実行すること

でハザードの発生を回避している。

WB	
MA	
EX	
	ID

○ 制御ハザード

制御の依存に起因するハザード

	1	2	3	4	5	6	7	8	9
BA label	IF	ID	EX	MA	WB				
add r2, r4, r5					IF	ID	EX	MA	WB

分岐命令がある場合、結果によって次に実行する命令が分岐条件によって異なるため、次の命令実行先がわかるまで停止していなければならない。分岐ハザードともいう。分岐命令がある限り逃れることのできないハザードである。

解消法

現在の CPU では、分岐命令があった場合その結果がどちらであると仮定 (予測) し、以降の命令を予め実行しておく、という動作をしている。この方法を分岐予測という。予測が当たっていれば滞り無く実行できるが、予測が外れていた場合、先んじて演算していた内容を全て捨て、分岐命令の直後から演算をやり直すため大幅なタイムロスが生じる。

現在においても分岐予測の精度向上は重要な課題の一つであり、研究が勧められている。

感想

とにかく時間がかかりました。実験もレポートも。

最近は電気店に行く度にパーツの所へ寄るようにしています。今はまだこんな物があるんだ、ぐらいしかわかりませんがいつかはきちんと理解できるようになりたいものです。

参考文献・引用文献

- LaTex -コマンド一覧
  http://www1.kiy.jp/~yoka/LaTeX/latex.html
- Wikipedia
  http://ja.wikipedia.org/wiki/
- IT 用語辞典 e-Words
  http://e-words.jp/