

コンピュータシステム基礎

情報工学科 阿部倫之

■ 担当教員： 阿部倫之

■ abe@neptune.kanazawa-it.ac.jp

■ 講義資料

配布資料、eシラバス

■ オフィスアワー

野々市キャンパス：月曜 5 限 2 1 号館 4 階教員控え室

八束穂キャンパス： 6 5 号館 2 1 0 室（要予約）

授業運営

- 日程： 1EP1, 1EP3
 - 第1週 (9/25)
 - 第2週 (10/2)
 - 第3週 (10/16)
 - 第4週 (10/23) レポート1 出題
 - 第5週 (10/30) 小テスト1 実施、レポート1 提出
 - 第6週 (11/5(月) 5限 補講 23.221教室)
 - 休講 11/6
 - 第7週 (11/13) レポート2 出題
 - 第8週 (11/20) 小テスト2 実施、レポート2 提出
 - 第9週 (11/27)
 - 第10週 (12/4) レポート3 出題

授業運営

- 日程（つづき） : 2EP1, 2EP3
 - 第1 1週（12/11）レポート3提出
 - 第1 2週（12/18）
 - 第1 3週（1/8）
 - 第1 4週（1/15）
 - 第1 5週（1/22） 期末試験、最終レポート提出
 - 第1 6週（1/29） 自己点検授業

小テスト2

- 実施日：平成30年11月20日（火） 授業時間
- 出題範囲：

第4章 コンピュータにおける計算の仕組み

- 4.1 メインメモリの構成
- 4.2 メインメモリの容量と番地
- 4.3 番地指定（アドレス指定）
- 4.4 メモリ内容の表示方法
- 4.5 命令と実行
 - 命令処理のサイクル
 - 命令の構造
 - 番地指定方式

※問題4.6

レポート2

- 提出日：平成30年11月20日（火）授業開始前
- レポート範囲：

補足問題集 3ページ ～ 5ページ （問題0.18は除く）

注意1）片面印刷で提出

注意2）ホッチキス止め不可（そのまま重ねて提出）

注意3）全ページにクラス番号、名前を記載

第8週

- レポート2 提出
- 講義（第5章）
- 小テスト2（復習15分、試験30分）

第5章 ALUで行う演算の16進表記

5.1 算術演算

- 加算、減算、比較（減算で代用）

5.2 論理演算

- 論理積、論理和、排他的論理和、論理否定

5.3 ビット列操作

- シフト： 左シフト、右シフト
- 回転シフト： 左回転シフト、右回転シフト

NT-ProcessorV1の概要

2 バイトワードマシン

- データワード長： 2バイト（16ビット）
- メインメモリ： 2バイト区切りでアドレス割付け
- 命令レジスタ（命令ワード）： 2バイト長
- 汎用レジスタ（データワード）： 2バイト長
- オペランドの指定方式： 2オペランド方式
- 演算命令の構成：
命令コード 第1オペランド, 第2オペランド

※演算結果は第1オペランドが指す場所に上書き

5.1 算術演算

5.1.1 加算

- 2の補数形式の加算

第1オペランド：被加数

第2オペランド：加数

※演算結果は第1オペランドが指す場所に上書き

※最上位桁が 8 ～ F のときに負数

- 正数どうしの加算例

$$\begin{array}{r} (0001\ 0010\ 0011\ 0100)_2 \\ + (0101\ 1010\ 1101\ 1111)_2 \\ \hline (0110\ 1101\ 0001\ 0011)_2 \end{array}$$

$$\begin{array}{r} (1234)_{16} \quad \leftarrow \text{第1オペランド} \\ + (5ADF)_{16} \quad \leftarrow \text{第2オペランド} \\ \hline (6D13)_{16} \end{array}$$

※オーバフロー： 加算によって符号を壊した場合は計算不能

・負数どうしの加算例

$$\begin{array}{r}
 (1110\ 0010\ 0011\ 0100)_2 \\
 + (1100\ 1010\ 1101\ 1111)_2 \\
 \hline
 1\ (1010\ 1101\ 0001\ 0011)_2 \\
 \uparrow \text{切捨て}
 \end{array}$$

$$\begin{array}{r}
 (E234)_{16} \\
 + (CADF)_{16} \\
 \hline
 1\ (AD13)_{16} \\
 \uparrow \text{切捨て}
 \end{array}$$

※オーバーフロー： 加算によって符号を壊した場合は計算不能

・正数と負数の加算例（切捨てありの例）

$$\begin{array}{r}
 (1110\ 0010\ 0011\ 0100)_2 \\
 + (0111\ 1010\ 1101\ 1111)_2 \\
 \hline
 1\ (0101\ 1101\ 0001\ 0011)_2 \\
 \uparrow \text{切捨て}
 \end{array}$$

$$\begin{array}{r}
 (E234)_{16} \\
 + (7ADF)_{16} \\
 \hline
 1\ (5D13)_{16} \\
 \uparrow \text{切捨て}
 \end{array}$$

正数どうしの場合と同様に計算する．16進表記で5桁目への桁上がりがある場合とない場合があり，ある場合は切り捨てる．加算結果が正数になる場合と負数になる場合がある．オーバーフローは発生しない．

5.1.2 減算

- 第2オペランドの減数（引く数）を「2の補数」に変換して加算

第1オペランド：被減数

第2オペランド：減数

※演算結果は第1オペランドが指す場所に上書き

- 負数から負数の減算・負数から正数の減算・正数から負数の減算
 - ・ 負数から負数の減算：「正数どうしの減算で被減数>減数の例」または「正数どうしの減算で被減数<減数の例」と同様に行う。オーバーフローは発生しない。
 - ・ 負数から正数の減算：「正数どうしの減算で被減数>減数の例」と同様に行う。結果が正数の場合はオーバーフローである。
 - ・ 正数から負数の減算：「正数どうしの減算で被減数<減数の例」と同様に行う。結果が負数の場合はオーバーフローである。

■ 演算結果に応じてフラグをセット

注) NT-Processor V1 では, ALUで加算または減算を行うと演算結果に応じて, 2ビットのコンディションコードレジスタCC (フラグレジスタともいう) に次の値が設定される. CCの内容は条件分岐命令で使われる.

演算結果が正数 : $CC = (10)_2 = (2)_{10}$

演算結果がゼロ : $CC = (00)_2 = (0)_{10}$

演算結果が負数 : $CC = (01)_2 = (1)_{10}$

演算結果がオーバーフロー : $CC = (11)_2 = (3)_{10}$

※CC : 2ビットのフラグレジスタ (条件コードレジスタ)

5.1.3 比較

- 第 1 オペランドと第 2 オペランドの絶対値の大小関係を判定

第 1 オペランドと第 2 オペランドの数値の符号を含めた絶対値の大小関係を判定し,
コンディションコードレジスタ CC (フラグレジスタともいう) に次の値を設定する.
CCの内容は条件分岐命令で使われる. 比較する数値は変更されない.

$> : CC = (10)_2 = (2)_{10}$

$= : CC = (00)_2 = (0)_{10}$

$< : CC = (01)_2 = (1)_{10}$

例 1 : 正数どうしの比較

第 1 オペランド $(3A57)_{16}$

第 2 オペランド $(1BDF)_{16}$

$> : CC = (10)_2 = (2)_{10}$

例 2 : 正数と負数の比較

第 1 オペランド (正数) $(3A57)_{16}$

第 2 オペランド (負数) $(8BDF)_{16}$

$< : CC = (01)_2 = (1)_{10}$

符号ビットを含めた絶対値比較であるため正数が小と判定される。

正数を大と判定するにはそれぞれの数値に $(8000)_{16}$ を加算して比較する。

第 1 オペランド (正数) $(BA57)_{16}$

第 2 オペランド (負数) $(0BDF)_{16}$

$> : CC = (10)_2 = (2)_{10}$

5.2 論理演算

- ・第1オペランドと第2オペランドを論理演算
- ・論理演算の結果は第1オペランドが指す場所に上書き

論理演算の種類

- ・論理積 (AND)
- ・論理和 (OR)
- ・排他的論理和 (XOR)
- ・論理否定 (NOT)

5.2.1 論理積 (AND)

・演算例

$(0001\ 0010\ 0011\ 0100)_2$	$(1234)_{16}$	←第1オペランド
<u>AND $(0101\ 1010\ 1101\ 1111)_2$</u>	<u>AND $(5ADF)_{16}$</u>	←第2オペランド
$(0001\ 0010\ 0001\ 0100)_2$	$(1214)_{16}$	

2進表記で第1オペランドのビットと第2オペランドの対応するビットがともに‘1’の場合にのみ演算結果のビットが‘1’になる。

・AND演算によるビット列マスク処理

2進表記で特定のビット列の値を‘0’にする操作をマスク処理という。次の例では第2オペランドに上位バイトがall ‘0’，下位バイトがall ‘1’のマスク値を設定してAND演算を行うことにより，第1オペランドの上位バイトをall ‘0’にしている

※下位8ビットのみを抽出

$(0001\ 0010\ 0011\ 0100)_2$	$(1234)_{16}$	←第1オペランド
<u>AND $(0000\ 0000\ 1111\ 1111)_2$</u>	<u>AND $(00FF)_{16}$</u>	←第2オペランド
$(0000\ 0000\ 0011\ 0100)_2$	$(0034)_{16}$	

5.2.2 論理和 (OR)

・演算例

$(0001\ 0010\ 0011\ 0100)_2$	$(1234)_{16}$	←第1オペランド
OR $(0101\ 1010\ 0100\ 0001)_2$	OR $(5A41)_{16}$	←第2オペランド
$(0101\ 1010\ 0111\ 0101)_2$	$(5A75)_{16}$	

2進表記で第1オペランドのビットと第2オペランドの対応するビットの少なくとも一方が‘1’の場合、演算結果のビットが‘1’になる。

(注: OR演算記号は“+”であるが算術加算と間違えるため, ここでは使っていない)

・OR演算によるビット列合成処理

2つのデータワードを部分的に組み合わせる操作をビット列合成処理という。次の例では第1オペランドの上位バイトをマスク処理でall ‘0’にしたデータワードと、第2オペランドの下位バイトをマスク処理でall ‘0’にしたデータワードとのOR演算により、下位バイトと上位バイトを組み合わせたデータワードを生成している。

$(0000\ 0000\ 0011\ 0100)_2$	$(0034)_{16}$	←第1オペランド
OR $(0101\ 1010\ 0000\ 0000)_2$	OR $(5A00)_{16}$	←第2オペランド
$(0101\ 1010\ 0011\ 0100)_2$	$(5A34)_{16}$	

5.2.3 排他的論理和 (XOR)

・演算例

$(0001\ 0010\ 0011\ 0100)_2$	$(1234)_{16}$	←第1オペランド
<u>XOR $(0101\ 1010\ 0100\ 0001)_2$</u>	<u>XOR $(5A41)_{16}$</u>	←第2オペランド
$(0100\ 1000\ 0111\ 0101)_2$	$(4875)_{16}$	

2進表記で第1オペランドのビットと第2オペランドの対応するビットが互いに‘1’と‘0’または‘0’と‘1’と異なるとき演算結果のビットが‘1’になる.

XOR演算では、2進表記でall ‘0’のデータワードとXORすると元の値が維持され、all ‘1’のデータワードとXORすると次のNOT演算と同様に各ビットの‘0’／‘1’が反転する.

$(1234)_{16}$	$(1234)_{16}$
<u>XOR $(0000)_{16}$</u>	<u>XOR $(FFFF)_{16}$</u>
$(1234)_{16}$	$(EDCB)_{16}$

5.2.4 論理否定 (NOT)

・演算例

$$\begin{array}{ll} \text{NOT } (0001\ 0010\ 0011\ 0100)_2 & \text{NOT } (1234)_{16} \quad \leftarrow \text{第2オペランド} \\ (1110\ 1101\ 1100\ 1011)_2 & (\text{EDCB})_{16} \quad \leftarrow \text{第1オペランド} \end{array}$$

2進表記で各ビットの‘0’／‘1’が反転する.

NOT演算は次に示す $(\text{FFFF})_{16}$ からの減算 (2進表記ではall ‘1’からの減算)と等価.

$$\begin{array}{r} (\text{FFFF})_{16} \\ - (1234)_{16} \\ \hline (\text{EDCB})_{16} \end{array}$$

数値データをNOT演算で‘0’／‘1’反転したものを1の補数 (16進表記では15の補数) という. 2の補数 (16進表記での16の補数) は1の補数に $(0001)_{16}$ を加算しても得られる.

5.3 ビット操作

第 2 オペランドに対してシフト操作を実施し、結果を第 1 オペランドが指す場所に上書き

5.3.1 1ビット左シフト（1ビット上位シフト）

$(b_{15} b_{14} \cdots b_1 b_0)_2$ ← 第2オペランド



$(b_{14} \cdots b_1 b_0 \text{ } \underline{0})_2$ ← 第1オペランド

※値は2倍になる

$CC = (0 \text{ } \underline{b_{15}})_2$

ビット列が1ビット上位シフト（左シフト）し、LSB（最下位ビット）に‘0’が入る。桁あふれした元のMSB（最上位ビット）はコンディションコードレジスタCCの下位ビットに入る。CCの上位ビットは‘0’になる。

16進表記でMSN（最上位ニブル）が $(3)_{16}$ 以下の正の数値データに対して1ビット上位シフトを行うと値が2倍になる。すなわち $(2)_{16}$ の乗算処理と等価。（注：MSNが $(7)_{16}$ の場合は2倍するとオーバーフローになる）

$(0001\ 0010\ 1010\ 0100)_2$



$(0010\ 0101\ 0100\ 1000)_2$

$(12A4)_{16}$



$(2548)_{16}$

$CC = (00)_2 = (0)_{10}$

5.3.2 1ビット右シフト（1ビット下位シフト）

$(b_{15} b_{14} \cdots b_1 b_0)_2$ ← 第2オペランド

↓

$(0 b_{15} b_{14} \cdots b_1)_2$ ← 第1オペランド

$CC = (0 b_0)_2$

※値は1/2倍になる

ビット列が1ビット下位シフト（右シフト）し，MSBに‘0’が入る．桁落ち（アンダーフロー）した元のLSBはコンディションコードレジスタCCの下位ビットに入る．CCの上位ビットは‘0’になる

正の数値データに対して1ビット下位シフトを行うと，値が1/2になる．すなわち $(2)_{16}$ による除算処理と等価．（以下ではCCは省略）

$(0010\ 0101\ 0100\ 1000)_2$

↓

$(0001\ 0010\ 1010\ 0100)_2$

$CC = (00)_2 = (0)_{10}$

$(2548)_{16}$

↓

$(12A4)_{16}$

5.3.3 1ビット左回転シフト（1ビット上位回転シフト）

$(\underline{b_{15}} b_{14} \cdots b_1 b_0)_2$ ← 第2オペランド

↓

$(b_{14} \cdots b_0 \underline{b_{15}})_2$ ← 第1オペランド

ビット列が1ビット上位シフト（左シフト）し，桁あふれした元のMSBがシフト後のLSBに入る．CC設定はない．

この操作は次のような数値データ操作に応用される．

$(\underline{1}000\ 0000\ 0000\ 0000)_2$

↓

$(0000\ 0000\ 0000\ 000\underline{1})_2$

$(8000)_{16}$

↓

$(0001)_{16}$

5.3.4 1ビット右回転シフト（1ビット下位回転シフト）

$(b_{15} b_{14} \cdots b_1 \underline{b_0})_2$ ←第2オペランド

↓

$(\underline{b_0} b_{15} b_{14} \cdots b_1)_2$ ←第1オペランド

ビット列が1ビット下位シフト（右シフト）し，桁落ちした元のLSBがシフト後のMSB

に入る．CC設定はない．

この操作は1ビット上位回転シフトの逆の数値データ操作に応用される．