目次

★1　条件分岐とフラグレジスタ

★1.1　分岐とは（前回の復習）

前回解説したように，CPU は命令をフェッチする度にプログラムレジスタ (PR) の値を自動的にインクリメントしていくので，通常の命令は，メモリに格納された番地の小さい方から順に実行されていく．しかし，PR の値を書きかえる命令があれば，その次の命令をどこからフェッチさせるかプログラムの側でコントロールすることができる．そのような命令を用いてプログラムの処理の流れを変えることを**分岐**という．分岐には，処理の流れを常に一定の場所に飛ばす**無条件分岐**と，何らかの条件に応じて分岐先を変える（分岐したりしなかったり）する**条件分岐**がある．いんちきアセンブリ言語では，JUMP 命令で無条件分岐させることができるのだった．

★1.2　条件分岐とフラグレジスタ

C 言語の if 文 や for 文，while 文のような処理を実現するためには，条件分岐を行えるようにしなければならない．一般的な CPU には，そのための仕組みとして，**フラグレジスタ** と呼ばれる特別なレジスタが備わっている．

フラグレジスタは，演算結果がゼロかどうか，正か負か，などを表すものであり，ALU が加減算等の演算を実行すると値がセットされる．フラグレジスタには様々なものがあるが，以下のものが代表的である．

ゼロフラグ（この授業では **ZF** と略記することにする）演算結果が 0 であるかどうかを表す (☆ 1)．結果が 0 なら 1，1 なら 0 となる．
サインフラグ（**SF** と略記）演算結果が負のときは 1，それ以外は 0(☆ 2)．
オーバーフローフラグ（**OF** と略記）演算結果がオーバーフローしたら 1，それ以外は 0．

いんちき計算機 II にはこれらの値を表すフラグレジスタが備わっており，ADDA, SUBA 等の算術加減算 (☆ 3) を実行すると値がセットされる．いんちき機械語にはこれらの値によって分岐する／しないを切り替える条件分岐命令（いんちきアセンブリ言語の JPL，JNZ 等，詳細は「いんちきアセンブリ言語命令集」参照）が用意されている．

☆ 1)「演算結果が 0 ⇔ALU の出力ビットが全て 0」だから，ALU の出力ビットの論理和 (OR) の否定 (NOT) が ZF となる．

☆ 2) 2 の補数表現を用いた符号付き整数同士の演算の場合，ALU 出力の最上位 bit が SF となる．

☆ 3) **算術加減算**: オペランドが符号付きの数であるとして行う加減算．いんちき計算機 II では 1 語 16bit で符号付き整数を表すので，-32768 から 32767 の範囲での演算となる．これに対して，いんちき計算機 II（というか CASL2）には，オペランドが表す bit 列をそのままの 2 進数として（0 から 65535 の数として）加減算を行う**論理加減算**の命令も存在する．

★1.3　if-then-else 型の条件分岐

C 言語の
`if(x == y)` なんたら; `else` かんたら;
という構造に対応するアセンブリ言語プログラムを考えてみよう（板書して説明します）．

Q1. 下記のプログラムを実行すると，記号番地 DOCCHI の値はいくつになるか．NEMREI を 20,21 にして実行した場合はどうか．

EX04IF

```
EX04IF   START
         LD    GR3,NEMREI
         LD    GR4,SEINEN
         LD    GR5,KODOMO
         SUBA  GR3,GR4
         JMI   HOGE
         SUBA  GR5,GR5
HOGE     ST    GR5,DOCCHI
         RET
NEMREI   DC    18
SEINEN   DC    20
KODOMO   DC    1
DOCCHI   DS    1
         END
```

★ 1.4 繰返し

「あれやこれやを n 回繰り返す」というような繰返し構造は，次のようないんちきアセンブリ言語プログラムで実現できる．ただし，

- 記号番地 N には繰返し回数が，ONE には 1 が格納されているとする
- あれやこれやの部分では GR1,GR2 は変化しない

と仮定している．

n 回の繰返し

```
         LD      GR1,N
         LD      GR2,ONE
LOOP     あれや
         これや
         SUBA    GR1,GR2
         JPL     LOOP
```

次は，C 言語における右のような構文に相当する繰返しの例である．上の例ではループ変数に相当するレジスタの値が N から 1 ずつ減っていくようになっていたのに対して，こちらの例では変数 i に対応するレジスタの値が 0 から 1 ずつ増えていく．for(i = 0; i < n; i++){ あれやこれや } という繰返しとよく似ているが少し動作が違うことに注意（↓や→のプログラムの場合「あれやこれや」を必ず一度は実行するが，for 文の場合一度も実行しないことがありえる）．

```
i = 0;
do{
   あれやこれや;
   i++;
}while(i < n);
```

do-while 型の繰返し

```
         LD      GR0,ZERO    ループ変数の初期化
         LD      GR1,N       ループ変数と比較する値をセット
         LD      GR2,ONE     繰り返し毎にループ変数に加える値をセット
LOOP     あれや               繰返し実行する命令
         これや               繰返し実行する命令
         ADDA    GR0,GR2     ループ変数の値を増やす
         CPA     GR0,GR1     比較
         JMI     LOOP         if (GR0) < (GR1) then goto LOOP
```

ただし，記号番地 ZERO, ONE, N には，0, 1, 変数 n に相当する数がそれぞれ格納されているとする．CPA という命令については「命令集」参照 (☆ 4).

☆ 4) SUBA を実行した場合と同じフラグレジスタの値を，実際に減算することなく（上記の例でいえば GR0 を変化させずに）セットすることができる命令である．

Q2. 下記のプログラムを実行すると，記号番地 HOGE の値はいくつになるか．N を 11 にして実行した場合はどうか．

EX04LOOP

```
EX04LOOP START
         LD    GR0,ZERO
         LD    GR1,ONE
         LD    GR2,N
         LD    GR7,ZERO
LOOP     ADDA  GR7,GR0
         ADDA  GR0,GR1
         CPA   GR0,GR2
         JMI   LOOP
         ST    GR7,HOGE
ZERO     DC    0
ONE      DC    1
N        DC    3
HOGE     DS    1
         END
```

★2 サブルーチン呼び出し

たいていの高級言語では，プログラムのあちこちで何度も同じ処理を行う必要がある場合には，その処理を 1 カ所にまとめて記述しておき（これを**サブルーチン**という），プログラム中からそれを呼び出して使えるようになっている．C 言語の関数もサブルーチンの一種と考えられる．いんちきアセンブリ言語でこのようなサブルーチンを作るにはどうしたらよいだろうか．

以下のプログラムは，JUMP 命令を用いてそれを実現しようと試みた例である(☆ 5)．9 行目から 11 行目までで「GR1 の 3 倍の値を GR7 にセットする」ようにして，記号番地 A の値の 3 倍を AX3 に，B の値の 3 倍を BX3 に書き込もうとしている．しかし，このプログラムは意図したようには動作しない．3 行目と 6 行目で SAMBAI(9 行目) に飛んで 3 倍の計算を行った後，それぞれ 4 行目と 7 行目に戻ってくれればよいのであるが，12 行目の JUMP 命令の戻り先は 4 行目に固定なのでそうならないからである．

☆ 5) このプログラムの 9 行目の LD 命令は，メモリからレジスタへデータを転送する他の LD 命令と違い，2 つのレジスタ間でデータ転送を行う 1 語の命令である．

EX04X3

```
 1  EX04X3   START
 2           LD    GR1,A
 3           JUMP  SAMBAI
 4  KOKO     ST    GR7,AX3
 5           LD    GR1,B
 6           JUMP  SAMBAI
 7  SOKO     ST    GR7,BX3
 8           RET
 9  SAMBAI   LD    GR7,GR1
10           ADDA  GR7,GR1
11           ADDA  GR7,GR1
12           JUMP  KOKO
13  A        DC    17
14  B        DC    -31
15  AX3      DS    1
16  BX3      DS    1
17           END
```

いんちきアセンブリ言語でサブルーチン呼び出しを実現するには，CALL 命令と RET 命令を用いる（板書して説明します）．