基礎コンピュータ工学
第５章 機械語プログラミング
(パート５：フラグと条件分岐)

https://github.com/tctsigemura/TecTextBook

本スライドの入手：

フラグ（1）

フラグ（C，S，Z）は計算結果の特徴を表す．
フラグ変化ありの命令を実行する度に値が変化する．
(教科書の本文，命令表を再度確認する．)

- *Z*（*Zero*）フラグ
  - Zero は「ゼロ」の意味．
  - 計算の結果がゼロにならなかった．

$$\begin{array}{rcl} & 0011\ 0010_2 & \\ + & 0011\ 0010_2 & \rightarrow \\ \hline \mathsf{Z}\ \boxed{0} & 0110\ 0100_2 & \end{array} \qquad \begin{array}{rr} & 50_{10} \\ + & 50_{10} \\ \hline & 100_{10} \end{array}$$

  - 計算の結果がゼロになった．

$$\begin{array}{rcl} & 0011\ 0010_2 & \\ - & 0011\ 0010_2 & \rightarrow \\ \hline \mathsf{Z}\ \boxed{1} & 0000\ 0000_2 & \end{array} \qquad \begin{array}{rr} & 50_{10} \\ - & 50_{10} \\ \hline & 0_{10} \end{array}$$

フラグ（2）

- *S*（*Sign*）フラグ
  - Sign はプラス・マイナスの「符号」の意味
  - 計算の結果を符号付き 2 進数と解釈すると正の値になった.

$$\begin{array}{rr} & 0011\ 0010_2 \\ + & 0011\ 0011_2 \\ \hline \mathsf{S}\ \boxed{0} & 0110\ 0101_2 \end{array} \quad \rightarrow \quad \begin{array}{rr} & 50_{10} \\ + & 51_{10} \\ \hline & 101_{10} \end{array}$$

  - 計算の結果を符号付き 2 進数と解釈すると負の値になった.

$$\begin{array}{rr} & 0011\ 0010_2 \\ - & 0011\ 0011_2 \\ \hline \mathsf{S}\ \boxed{1} & 1111\ 1111_2 \end{array} \quad \rightarrow \quad \begin{array}{rr} & 50_{10} \\ - & 51_{10} \\ \hline & -1_{10} \end{array}$$

- *S* フラグは符号付き 2 進数と考えたときの「負」の意味
- 計算結果の最上位ビットと同じ値になる．→ゼロは「正」とみなす.

フラグ（3）

- *C*（*Carry*）フラグ
  - Caryy は「桁を繰り上げる」の意味
  - 足し算（ADD）で桁上げが起きる.
    - 足し算で最上位桁からの桁上げがない場合

$$\begin{array}{rcl} & 0111\ 1111_2 & \\ + & 0000\ 0001_2 & \\ \hline \mathsf{C}\ \boxed{0} & 1000\ 0000_2 & \end{array} \quad \rightarrow \quad \begin{array}{rr} & 127_{10} \\ + & 1_{10} \\ \hline & 128_{10} \end{array}$$

    - 足し算で最上位桁からの桁上げがあった場合（オーバーフロー）

$$\begin{array}{rcl} & 1111\ 1111_2 & \\ + & 0000\ 0001_2 & \\ \hline \mathsf{C}\ \boxed{1} & 0000\ 0000_2 & \end{array} \quad \rightarrow \quad \begin{array}{rr} & 255_{10} \\ + & 1_{10} \\ \hline & 0_{10} \end{array} \quad ?$$

フラグ（4）

- *C*（*Carry*）フラグ（Borrow の意味を代用）
  - Borrow は「桁を借りる」の意味
  - 引き算（SUB）で桁借りが起こる
    - 引き算で最上位桁で桁借りがない場合

$$\begin{array}{rr|l} & & 1111\ 1111_2 \\ - & & 0000\ 0001_2 \\ \hline \mathrm{C} & \boxed{0} & 1111\ 1110_2 \end{array} \quad \rightarrow \quad \begin{array}{rl} & 255_{10} \\ - & 1_{10} \\ \hline & 254_{10} \end{array}$$

    - 引き算で最上位桁で桁借りがあった場合（負にオーバーフロー）

$$\begin{array}{rr|l} & & 0000\ 0000_2 \\ - & & 0000\ 0001_2 \\ \hline \mathrm{C} & \boxed{1} & 1111\ 1111_2 \end{array} \quad \rightarrow \quad \begin{array}{rl} & 0_{10} \\ - & 1_{10} \\ \hline & 255_{10} \end{array} \quad ?$$

- *C* フラグは，符号なし 2 進数と考えたときのオーバーフローの意味

ジャンプ命令（７種類）

無条件ジャンプ命令： プログラムの流れを指定のアドレスに飛ばす.

- *JMP*（*Jump*）命令：いつもジャンプする.

条件ジャンプ命令： ある条件のときだけジャンプする.

- *JZ*（*Jump on Zero*）命令：$Z = 1$ ならジャンプ
- *JC*（*Jump on Carry*）命令：$C = 1$ ならジャンプ
- *JM*（*Jump on Minus*）命令：$S = 1$ ならジャンプ
- *JNZ*（*Jump on Not Zero*）命令：$Z = 0$ ならジャンプ
- *JNC*（*Jump on Not Carry*）命令：$C = 0$ ならジャンプ
- *JNM*（*Jump on Not Minus*）命令：$S = 0$ ならジャンプ

JZ（Jump on Zero）命令

Z フラグが 1 なら（計算結果が 0 なら）ジャンプする.

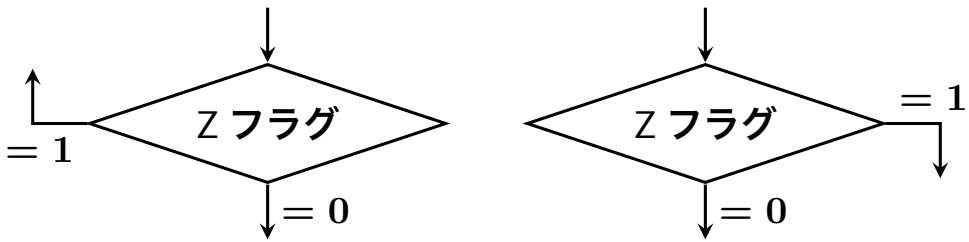
フラグ： 変化しない.

ニーモニック： `JZ EA`　　（`if(Z=1) PC ← EA`）

命令フォーマット： 2 バイトの長さを持つ.

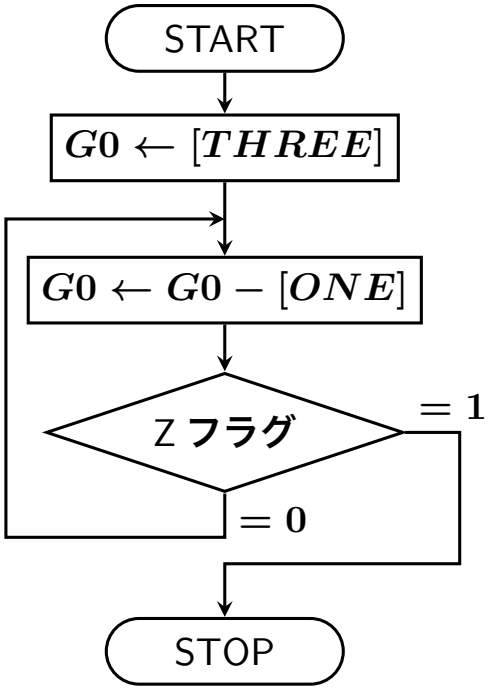
第 1 バイト		第 2 バイト
OP	GR XR	
1010_2	01_2 *XR*	*aaaa aaaa*

フローチャート： ある程度，自由にアレンジしてよい.



JZ 命令の使用例

ループを 3 回，繰り返すプログラム



番地	機械語	ラベル	ニーモニック	
00	10 09		LD	G0,THREE
02	40 0A	LOOP	SUB	G0,ONE
04	A4 08		JZ	STOP
06	A0 02		JMP	LOOP
08	FF	STOP	HALT	
09	03	THREE	DC	3
0A	01	ONE	DC	1

- 演習（1）：ステップモードで実行をトレースしてみる．

JC（Jump on Carry）命令

C フラグが 1 なら（オーバーフローなら）ジャンプする.

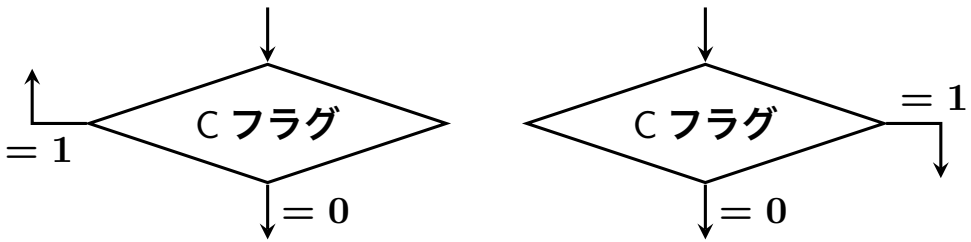
フラグ： 変化しない.

ニーモニック： `JC EA`　　(`if(C=1) PC ← EA`)

命令フォーマット： 2 バイトの長さを持つ.

第 1 バイト		第 2 バイト
OP	GR XR	
1010_2	10_2 *XR*	*aaaa aaaa*

フローチャート： ある程度，自由にアレンジしてよい.



JM（Jump on Minus）命令

S フラグが 1 なら（負なら）ジャンプする.

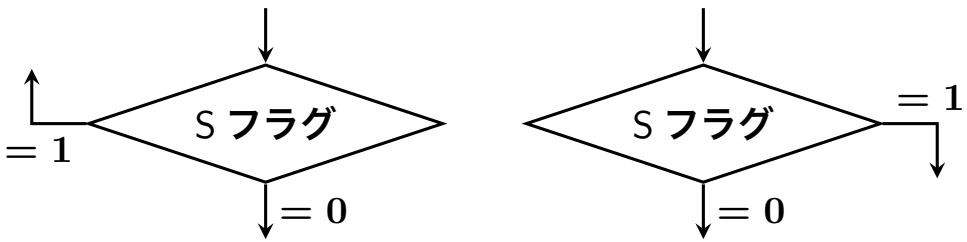
フラグ： 変化しない.

ニーモニック： `JM EA`　　（`if(S=1) PC ← EA`）

命令フォーマット： 2 バイトの長さを持つ.

第 1 バイト		第 2 バイト
OP	GR XR	
1010_2	11_2 *XR*	*aaaa aaaa*

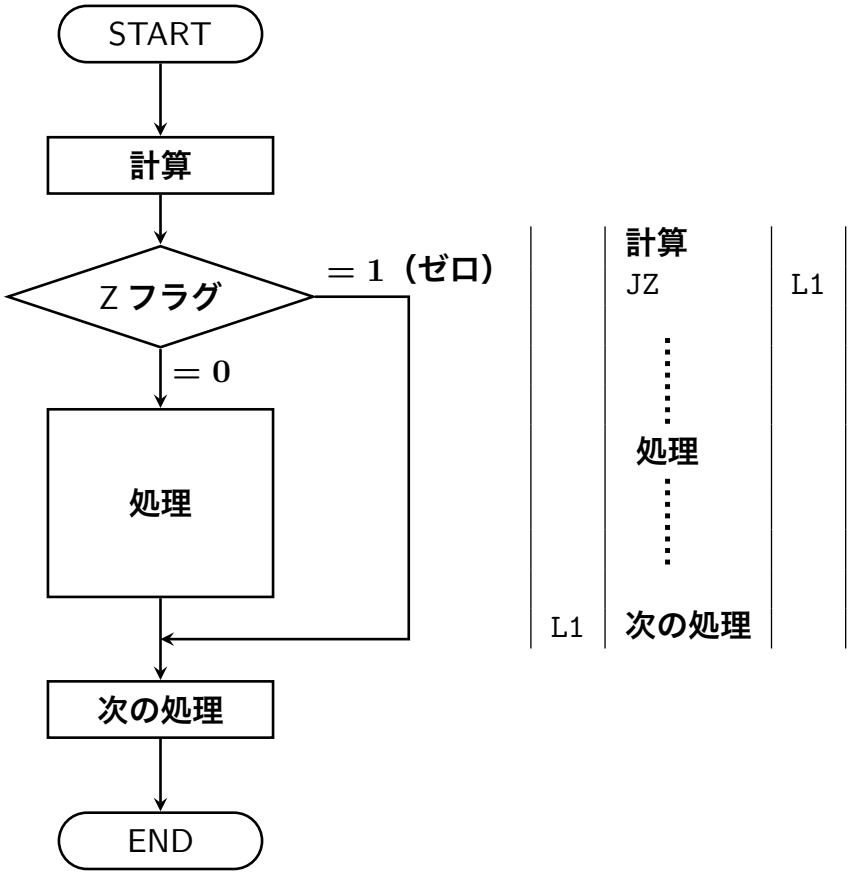
フローチャート： ある程度，自由にアレンジしてよい.



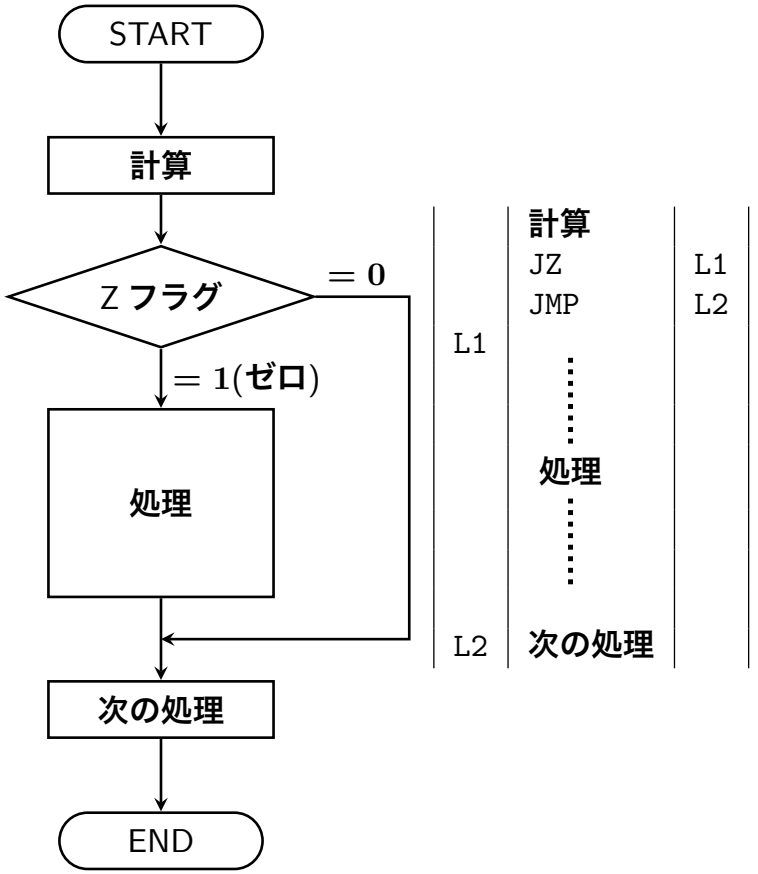
条件判断 1

計算結果により処理をするかしないか変化する例

計算結果がゼロなら「処理」しない



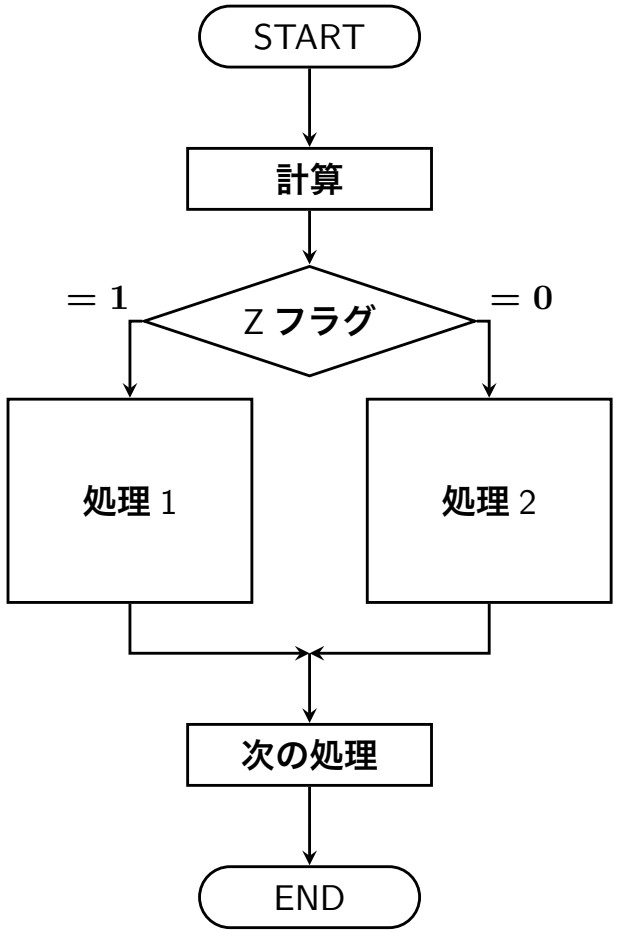
計算結果がゼロなら「処理」する



条件判断２

計算結果によりどちらかの処理をする例

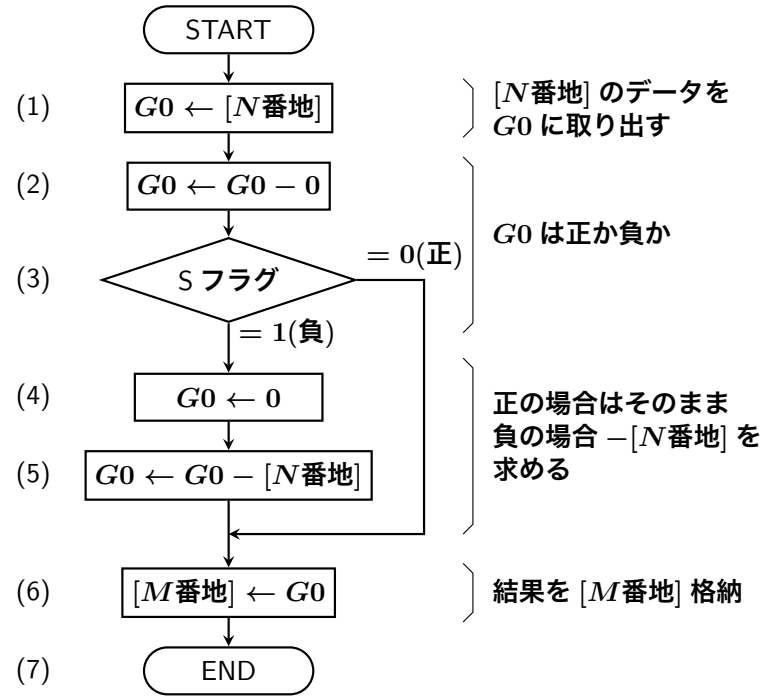
計算結果がゼロなら「処理１」，そうでなければ「処理２」をする




```
        計算
        JZ          L1
        ⋮
        処理 2
        ⋮
        JMP         L2
        ⋮
L1      処理 1
        ⋮
L2      次の処理
```

条件判断の例

絶対値を求めるプログラム（例題 5-1）



注意：$[N番地]$ は、N 番地に格納されているデータのこと

番地	機械語	ラベル	ニーモニック	
00	10 10	START	LD	G0,N
02	40 0F		SUB	G0,ZERO
04	AC 08		JM	L1
06	A0 0C		JMP	L2
08	10 0F	L1	LD	G0,ZERO
0A	40 10		SUB	G0,N
0C	20 11	L2	ST	G0,M
0E	FF		HALT	
0F	00	ZERO	DC	0
10	FF	N	DC	-1
11	00	M	DS	1

- 演習（２）：ステップモードで実行をトレースしてみる.

まとめ

学んだこと

- フラグ（Carry，Zero，Sign）
- 条件ジャンプ命令（JZ，JC，JM）
- 条件判断

演習（宿題）

- 飽和演算：計算結果が最大値または最小値を超えそうになった時，計算結果を最大値または最小値に留める演算方式
- TeC の符号なし 2 進数を用いて表現できる最大値は 255 である．
- 足し算結果が 255 を超える（オーバーフローする）かもしれない．
- オーバーフローが発生したら計算結果を 255 に訂正するようにする．
- 以上のような足し算プログラムを作る．