電子計算機 **3E**

3E　2003.11.21

後期中間テストについて

(1) テストは、来週の授業中に実施します。
- 11 月 28 日(金)2 時間目 9:40〜10:30
- テスト範囲は、教科書の p.86 まで
- これまでの学習をまとめたものを、次ページ以降に示すので、それをよく学習すること。

(2) 60 点以下の者は再テストを実施します。
- 再テストの日時、および受験者は追って連絡する。

1．CASL II の命令の種類

CASL II の命令は、3 種類あります。

アセンブラ命令

教科書の P.28～P.35 で説明している非実行文とかかれているものです。アセンブラーという変換プログラムに対して、いろいろな指示を行う命令です。COMET II の CPU の動作の指示は行いません。したがって、この命令は機械語に変換されて特定のビットパターン(1 と 0 の組み合わせ)に変換されることはありません。

CASL II には、次の 4 個のアセンブラ命令があります。

START	プログラムの先頭を定義 プログラムの実行開始番地を定義 他のプログラムで参照する入口名を定義
END	プログラムの終わりを明示
DC	領域を確保
DS	定数を定義

機械語命令

教科書の P.40～P.82 で説明しています。この命令は、COMET II の CCPU の動作の指示を行います。機械語命令は、アセンブラーにより特定のビットパターンの機械語に変換されてます。実行時には、そのビットパターンが主記憶装置に格納されます。ビットパターンへの変換は、教科書の P.213 に書かれています。

CASL II には、以下の 28 個のアセンブラ命令があります(教科書の P.203)。

ロード、ストア	LD ST LAD
算術、論理演算	ADDA ADDL SUBA SUBL AND OR XOR
比較演算	CPA CPL
シフト演算	SLA SRA SLL SRL
分岐	JPL JMI JNZ JZE JOV JUMP
スタック操作	PUSH POP
コール、リターン	CALL RET
その他	SVC NOP

マクロ命令

教科書の P.83～P.86 で説明しています。マクロ命令とは、特定の機能を果たすためのいくつかの命令の集まりに名前を付けたものです。この名前を指定するだけで、これらの命令の集まりが実行されます。これにより、頻繁に使われる定形的な命令群をマクロ命令にすることにより、同じようなプログラムをいちいち書くことを省くことができます。

多くの命令から構成されるため、アセンブラーにより変換されるビットパターンは非常に多くなります。

CASL II には、以下の 4 個のマクロ命令があります(教科書の P.203)。

IN	入力装置(キーボード)から、文字データを読み込む
OUT	出力装置(ディスプレイ)に、文字データを書き込む
RPUSH	GR の内容を、GR1、GR2、・・・、GR7 の順でスタックに格納
RPOP	スタックの内容を GR7、GR6、・・・、GR1 の順で GR に格納

2．ラベルについて

記述の約束は以下の通りです。

- プログラムのロジックでラベルが不要な場合は、記述しなくても良いです。
- ラベルは、8 文字以内です。先頭はアルファベットの大文字、2 文字以降はアルファベットの大文字、数字いずれでも良いです。
- 必ず先頭(第 1 文字)から始めます。第 1 文字が空白の場合は、ラベル名は無いものみなされ、命令コードと解釈されます。
- 汎用レジスタの名前の GR0 から GR7 は予約語であり、ラベル名として使用できません。命令コードのオペランドで、ラベルなのかレジスタなのか区別できなくなります。

ラベルは、その記述する位置から FORTRAN の文番号にも似ています。あるいは、変数名にも似ています。実際、FORTRAN の文番号や変数名のような使われ方をします。しかし実際には、これはアドレスをあらわします。アドレスは、命令に従い、以下のようになっています。

- 機械語命令のラベルの場合は、その機械語命令が格納されている 2 語分の領域のうち、その先頭アドレスを表します。実際のプログラムでは、ジャンプ命令とともに使われ、そのアドレスに制御が移ります。FORTRAN の GO TO 分でその文番号に制御が移るのと同じです。
- DC 命令[1]の場合、ラベルは定数が格納されている領域[2]の先頭アドレスを示します。使い方は FORTRAN の変数名に似ていますが、実態はアドレスです。
- DS 命令[3]の場合、ラベルはこの命令によって確保されている主記憶の領域[4]のの先頭アドレスを示します。
- IN や OUT のマグロ命令の場合は、ラベルは複数の命令群のうちの先頭の命令が格納されているアドレスを示します。

ラベルがアドレスを表すことが理解できれば、簡単ですよね。常識通りに解釈すればよいのです。

重要なポイント

以前学習した通りアセンブラのプログラムは、主記憶装置(メインメモリー)の中に格納されているデータを処理(いろいろな演算)します。また、プログラムの命令も主記憶装置に格納されています。主記憶装置に格納されているデータや命令にアクセスする場合、主記憶装置のアドレスを指定することになります。したがって、アセンブラでは、アドレスが重要になり、プログラマーは意識しなくてはなりません。

高級言語の場合、アドレスに関してはコンパイラーが勝手に処理をしてくれます。例えば、FORTRAN で変数を使った場合、プログラマーがその変数のアドレスに注意を払う必要はありません。これは、コンパイラーが変数とアドレスの関係の表を持っており、それに従い、上手にマシン語に変換してくれるからです。

アセンブラのでは、コンパイラーの代わりにプログラマーが変数とアドレスの関係を考えなくてはなりません。

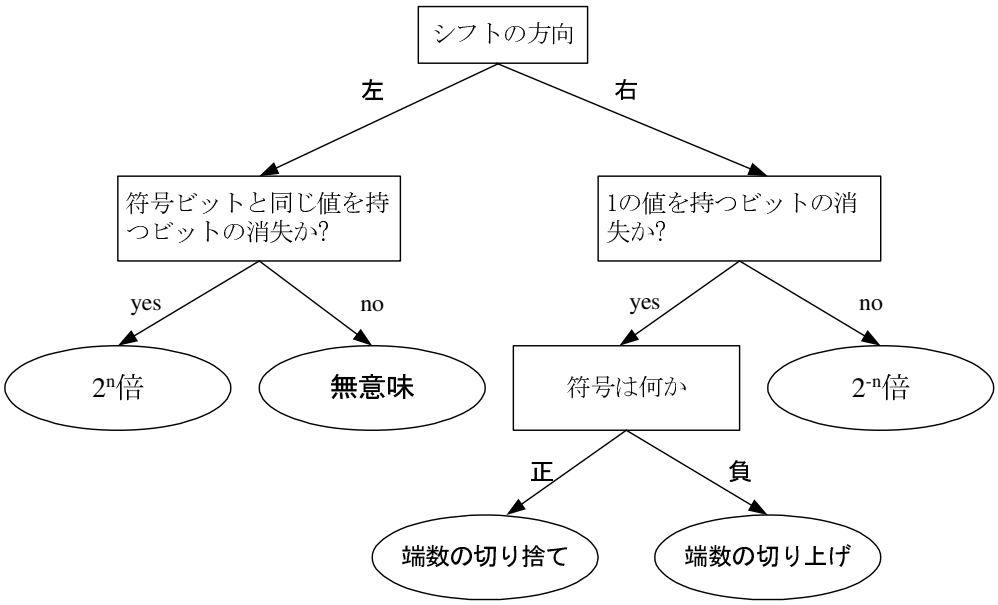
---

[1] Define Constant　データを定義する。実際には、実行時、主記憶装置にデータを格納する。
[2] 1 語の場合もあれば、複数語の場合もある。
[3] Define Storage 主記憶装置にデータを格納する領域を確保する。
[4] 1 語の場合もあれば、複数語の場合もある。

3．算術シフト命令について

算術シフトの実行結果は、以下のようになる(前回の授業で議論)。

算術シフト命令(SLA, SRA)の実行結果



4.命令一覧

4.1 アセンブラ命令

機能	書式	機能	FR レジスタの動作
プログラム開始	[ラベル] START [実行開始番地]	プログラムの開始を示す。プログラムの最初に、必ず書かなくてはならない。	
プログラム終了	END	プログラムの終わりを示す。ラベルは使えない。プログラムの最後に、必ず書かなくてはならない。	
10 進定数格納	[ラベル] DC n	10 進定数をラベルのアドレスに格納	
16 進定数格納	[ラベル] DC #h	16 進定数をラベルのアドレスに格納	
文字定数格納	[ラベル] DC ’文字列’	文字列をラベルのアドレスから格納	
アドレス定数格納	[ラベル] DC ラベル名	ラベル名で示すアドレスを格納	
領域の確保	[ラベル] DS n	ラベル名で示すアドレスから n 語領域を確保	

注意

・アセンブラ命令ではフラグレジスタの値はセットされることはない。これは、アセンブラ命令はプログラム実行には動作しないためである。

4.2 機械語命令

機能	書式	機能	FR レジスタの動作
ロード	[ラベル] LD r1,r2	レジスタ r2 の値をレジスタ r1 にコピーする。	OF: = 0(常にゼロが設定される) SF: = 1(負のとき、ビット番号 15 が 1) = 0(正のとき、ビット番号 15 が 0) ZF: = 1(値がゼロの時、全てのビットが 0) = 0(上記以外)
	[ラベル] LD r,adr[,x]	アドレス adr[,x]の主記憶装置の内容をレジスタ r にコピーする。	
ストア	[ラベル] ST r,adr[,x]	レジスタ r の内容を主記憶装置のアドレス adr[,x]にコピーする	変化なし
ロードアドレス	[ラベル] LAD r,adr[,x]	主記憶装置のアドレス値 adr[,x]をレジスタ r にコピーする。	変化なし
算術加算	[ラベル] ADDA r1,r2	レジスタ r1 と r2 の符号付加算 r1←r1+r2	OF: = 1(演算の結果が-32768～32767 の範囲外) = 0(上記以外) SF: = 1(演算の結果が負のとき、ビット番号 15 が 1) = 0(演算の結果が正のとき、ビット番号 15 が 0) ZF: = 1(演算の結果がゼロの時、全てのビットが 0) = 0(上記以外)
	[ラベル] ADDA r,adr[,x]	レジスタ r と主記憶装置(アドレス adr[,x])の内容を符号付加算　r←r+adr[,x]の内容	
算術減算	[ラベル] SUBA r1,r2	レジスタ r1 と r2 の符号付減算 r1←r1-r2	
	[ラベル] SUBA r,adr[,x]	レジスタ r と主記憶装置(アドレス adr[,x])の内容を符号付減算　r←r-adr[,x]の内容	
論理加算	[ラベル] ADDL r1,r2	レジスタ r1 と r2 の符号無加算 r1←r1+r2	OF: = 1(演算の結果が 0～65535 の範囲外) = 0(上記以外) SF: = 1(演算の結果が負のとき、ビット番号 15 が 1) = 0(演算の結果が正のとき、ビット番号 15 が 0) ZF: = 1(演算の結果がゼロの時、全てのビットが 0) = 0(上記以外)
	[ラベル] ADDL r,adr[,x]	レジスタ r と主記憶装置(アドレス adr[,x])の内容を符号無加算　r←r+adr[,x]の内容	
論理減算	[ラベル] SUBL r1,r2	レジスタ r1 と r2 の符号無減算 r1←r1-r2	
	[ラベル] SUBL r,adr[,x]	レジスタ r と主記憶装置(アドレス adr[,x])の内容の符号無減算　r←r-adr[,x]の内容	

機能	書式	機能	FR レジスタの動作
論理積	[ラベル] AND r1,r2	レジスタ r1 とレジスタ r2 のビット毎の論理積を計算。結果は r1 に格納	OF: = 0(常にゼロが設定される) SF: = 1(演算の結果が負のとき、ビット番号 15 が 1) = 0(演算の結果が正のとき、ビット番号 15 が 0) ZF: = 1(演算の結果がゼロの時、全てのビットが 0) = 0(上記以外)
	[ラベル] AND r,adr[,x]	レジスタ r と主記憶装置(アドレス adr[,x])の内容のビット毎の論理積を計算。結果は r に格納	
論理和	[ラベル] OR r1,r2	レジスタ r1 とレジスタ r2 のビット毎の論理和を計算。結果は r1 に格納	
	[ラベル] OR r,adr[,x]	レジスタ r と主記憶装置(アドレス adr[,x])の内容のビット毎の論理和を計算。結果は r に格納	
排他的論理和	[ラベル] XOR r1,r2	レジスタ r1 とレジスタ r2 のビット毎の排他的論理和を計算。結果は r1 に格納	
	[ラベル] XOR r,adr[,x]	レジスタ r と主記憶装置(アドレス adr[,x])の内容のビット毎の排他的論理和を計算。結果は r に格納	
算術比較	[ラベル] CPA r1,r2	レジスタ r1 とレジスタ r2 を符号付の数として比較。結果は FR レジスタに格納	2 つの数字の比較(以下の演算)により、FR が設定される。 r1-r2 r-adr[,x]の内容 OF: = 0(常にゼロが設定される) SF: = 1(演算の結果が負のとき、ビット番号 15 が 1) = 0(演算の結果が正のとき、ビット番号 15 が 0) ZF: = 1(演算の結果がゼロの時、全てのビットが 0) = 0(上記以外)
	[ラベル] CPA r,adr[,x]	レジスタ r1 と主記憶装置(アドレス adr[,x])の内容を符号付の数として比較。結果は FR レジスタに格納	
論理比較	[ラベル] CPL r1,r2	レジスタ r1 とレジスタ r2 を符号無の数として比較。結果は FR レジスタに格納	
	[ラベル] CPL r,adr[,x]	レジスタ r1 と主記憶装置(アドレス adr[,x])の内容を符号無の数として比較。結果は FR レジスタに格納	

機能	書式	機能	FR レジスタの動作
算術左シフト	[ラベル] SLA r,adr[,x]	レジスタ r の内容を符号ビットを除き、adr[,x]の番地そのもの分、各ビットを左へシフト。空いたビットには 0 が入る。	OF: = レジスタから、最後に送り出されたビットの値 SF: = 1(ビット番号 15 が 1 のとき) = 0(ビット番号 15 が 0 のとき) ZF: = 1(全てのビットが 0 のとき) = 0(上記以外)
算術右シフト	[ラベル] SRA r,adr[,x]	レジスタ r の内容を符号ビットを除き、adr[,x]の番地そのもの分、各ビットを右へシフト。空いたビットには符号ビットと同じ値が入る。	
論理左シフト	[ラベル] SLL r,adr[,x]	レジスタ r の内容を、adr[,x]の番地そのもの分、各ビットを左へシフト。空いたビットには 0 が入る。	OF: = レジスタから、最後に送り出されたビットの値 SF: = 1(ビット番号 15 が 1 のとき) = 0(ビット番号 15 が 0 のとき) ZF: = 1(全てのビットが 0 のとき) = 0(上記以外)
論理右シフト	[ラベル] SRL r,adr[,x]	レジスタ r の内容を、adr[,x]の番地そのもの分、各ビットを右へシフト。空いたビットには 0 が入る。	
正分岐	[ラベル] JPL adr[,x]	FR レジスタの SF と ZF の両方が 0 の時、adr[,x]のアドレスへ分岐(実行が移動)する。	変化なし
負分岐	[ラベル] JMI adr[,x]	FR レジスタの SF が 1 の時、adr[,x]のアドレスへ分岐(実行が移動)する。	
非零分岐	[ラベル] JNZ adr[,x]	FR レジスタの ZF が 0 の時、adr[,x]のアドレスへ分岐(実行が移動)する。	
零分岐	[ラベル] JZE adr[,x]	FR レジスタの ZF が 1 の時、adr[,x]のアドレスへ分岐(実行が移動)する。	
オーバーフロー分岐	[ラベル] JOV adr[,x]	FR レジスタの OF が 1 の時、adr[,x]のアドレスへ分岐(実行が移動)する。	
無条件分岐		無条件に、adr[,x]のアドレスへ分岐(実行が移動)する。	変化なし

機能	書式	機能	FR レジスタの動作
プッシュ	[ラベル] PUSH adr[,x]	スタックポインタを 1 減らして、その番地へadr[,x]のアドレスを格納する。	変化なし
ポップ	[ラベル] POP r	スタックポインタで示されている番地の内容をレジスタ r にコピーする。そしてスタックポインタの値を 1 加算する。	
コール	[ラベル] CALL adr[,x]	サブルーチンを呼び出す。呼び出すサブルーチンは、アドレス adr[,x]から始まる。	変化なし
リターン	[ラベル] RET	サブルーチンからメインルーチンへ戻る。	
スーパーバイザーコール	[ラベル] SVC adr[,x]	OS の機能を呼び出す。IN や OUT のマクロ命令で使われている。	不定　OS に依存する
ノーオペレーション	[ラベル] NOP	何もしない命令です。	変化なし

4.3 マクロ命令

機能	書式	機能	FR レジスタの動作
入力命令	[ラベル] IN ラベル 1,ラベル 2	入力領域(ラベル 1)に文字データを入れる。入力文字長は、ラベル 2 に入る。	不定　OS に依存する
出力命令	[ラベル] OUT ラベル 1,ラベル 2	出力領域(ラベル 1)の文字データを出力する。出力文字長は、ラベル 2 で示す。	不定　OS に依存する
レジスタの待避	[ラベル] RPUSH	汎用レジスタ GR1～GR7 の内容を、GR1→GR7 の順序でスタック領域に格納。	不定　OS に依存する
レジスタの復元	[ラベル] RPOP	スタック領域の内容を、GR7→GR1 の順序で汎用レジスタに格納。	不定　OS に依存する