これまで、命令という抽象的な言葉を使って来ましたが、コンピュータが命令として直接理解できるのは**機械語**で、どのプログラム言語を使ってプログラムを作成しても最終的には機械語に変換され、コンピュータ上で実行されます。これから習う C 言語も例外ではありません。コンピュータが登場した当時は機械語を使ってプログラムを作成していましたが、**コンピュータよりの言語** (**低水準言語**; **低レベル言語**) であったためプログラムの作成・変更・追加・削除が楽ではありませんでした。また、コンピュータの種類が異なるとあらためてプログラムを作成する必要がありました。その後、機械語と 1 対 1 に対応した**アセンブラ言語** (**アセンブリ言語**) が登場しました。この言語は、人間が理解しやすい記述方法やプログラムを実行する直前に機械語に変換するといった仕組みが取り入れられ、機械語の欠点が改善されました。しかしながら、この言語も中央処理装置の違いによって仕様が異なるなど、欠点を残したままでした。その後も、コンピュータの進歩やプログラムの構文解析技術の発達などにより、様々なアイデアと改良が加えられ、次々と新しい**人間よりの言語** (**高水準言語**; **高レベル言語**; **高級言語**) が誕生しました (詳細は 5.5 節)。

ここで、コンピュータの動作の仕組みをより深く理解するために、アセンブラ言語 (機械語) について見ておきましょう (ハードウェアからソフトウェアへの橋渡しを行いましょう)。具体的には、独立行政法人 情報処理推進機構 (IPA) が情報処理技術者試験用に策定した COMET II という仮想コンピュータ上で動作する CASL II というアセンブラ言語を用いることにします。CASL II (アセンブラ言語) には、コンピュータが動作するために最低限必要な表 4.4 のような命令が定義されています。なお、COMET II と CASL II の仕様については、独立行政法人 情報処理推進機構の情報処理技術者試験のホームページ (http://www.jitec.ipa.go.jp) から「試験要綱・シラバス・過去問題 など」を選択し、リンク先の下の方にある「試験で使用する情報処理用語・プログラム言語など」を参照してください[15]。

命令の種類	命令の種類 (詳細)
ロード, ストア, ロードアドレス	ロード, ストア, ロードアドレス
算術演算, 論理演算	算術加算, 論理加算, 算術減算, 論理減算, 論理積, 論理和, 排他的論理和
比較演算	算術比較, 論理比較
シフト演算	算術左シフト, 算術右シフト, 論理左シフト, 論理右シフト
分岐	正分岐, 負分岐, 非零分岐, 零分岐, オーバフロー分岐, 無条件分岐
スタック操作	プッシュ, ポップ
コール, リターン	コール, リターン
その他	スーパバイザコール, ノーオペレーション

表 4.4: アセンブラ言語 CASL II の命令

では、具体的な CASL II (アセンブラ言語) によるプログラムをいくつか挙げ、そのプログラムを考察することでアセンブラ言語をより深く理解しましょう。

[15] 具体的に CASL II (アセンブラ言語) を学習したい人は、独立行政法人 (IPA) の情報処理技術者試験のホームページ (http://www.jitec.ipa.go.jp) から CASL II のシミュレータをダウンロードすることができます。

例 1 このアセンブラ言語によるプログラムは、

$$1 + 2 \quad (= 3)$$

を計算するプログラムである。

ラベル	命令コード	オペランド	
EXAM	START		プログラム開始 (OS から処理を継続)
	LD	GR0,DATA1	レジスタ GR0 に定数 DATA1 の値を代入
	ADDA	GR0,DATA2	レジスタ GR0 の値に定数 DATA2 の値を加え、GR0 に代入
	ST	GR0,ANS	レジスタ GR0 の値を変数 ANS に書き出す
	RET		命令終了 (OS に処理を返す)、以下の部分はデータ
DATA1	DC	#0001	定数名DATA1 の定数に 16 進数 1 を設定
DATA2	DC	#0002	定数名DATA2 の定数に 16 進数 2 を設定
ANS	DS	1	変数名ANS の変数に 1 語 (16 ビット) を確保
	END		プログラム終了

例 2 このアセンブラ言語によるプログラムは、

$$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 \quad (= 55)$$

を計算するプログラムである。

ラベル	命令コード	オペランド	
PROG	START		プログラム開始 (OS から処理を継続)
	LD	GR0,NUM0	レジスタ GR0 に定数 NUM0 の値を代入
	LD	GR1,NUM0	レジスタ GR1 に定数 NUM0 の値を代入
LOOP	ADDA	GR1,NUM1	レジスタ GR1 の値に定数 NUM1 の値を加え、GR1 に代入
	ADDA	GR0,GR1	レジスタ GR0 の値にレジスタ GR1 の値を加え、GR0 に代入
	CPA	GR1,NUM2	レジスタ GR1 の値と定数 NUM2 の値を比較 (フラグ変化を伴う)
	JMI	LOOP	サインフラグ SF が真 (1) ならばラベル LOOP にジャンプ
	ST	GR0,ANS	レジスタ GR0 の値を変数 ANS に書き出す
	RET		命令終了 (OS に処理を返す)、以下の部分はデータ
NUM0	DC	#0000	定数名NUM0 の定数に 16 進数 0 を設定
NUM1	DC	#0001	定数名NUM1 の定数に 16 進数 1 を設定
NUM2	DC	#000A	定数名NUM2 の定数に 16 進数 A を設定
ANS	DS	1	変数名ANS の変数に 1 語 (16 ビット) を確保
	END		プログラム終了

例題 1　以下のアセンブラ言語によるプログラムを実行したとき、変数 ANS に格納されている値を 10 進数で答えなさい。

ラベル	命令コード	オペランド	
FFUN	START		プログラム開始 (OS から処理を継続)
	LD	GR0,NUM1	レジスタ GR0 に定数 NUM1 の値を代入
	LD	GR1,NUM1	レジスタ GR1 に定数 NUM1 の値を代入
	LD	GR2,NUM1	レジスタ GR2 に定数 NUM1 の値を代入
	LD	GR3,NUM0	レジスタ GR3 に定数 NUM0 の値を代入
LOOP	ADDA	GR0,GR2	レジスタ GR0 の値にレジスタ GR2 の値を加え、GR0 に代入
	ADDA	GR3,NUM1	レジスタ GR3 の値に定数 NUM1 の値を加え、GR3 に代入
	LD	GR2,GR1	レジスタ GR2 にレジスタ GR1 の値を代入
	LD	GR1,GR0	レジスタ GR1 にレジスタ GR0 の値を代入
	CPA	GR3,NUM2	レジスタ GR3 の値と定数 NUM2 の値を比較 (フラグ変化を伴う)
	JNZ	LOOP	ゼロフラグ ZF が偽 (0) ならばラベル LOOP にジャンプ
	ST	GR0,ANS	レジスタ GR0 の値を変数 ANS に書き出す
	RET		命令終了 (OS に処理を返す)、以下の部分はデータ
NUM0	DC	0	定数名NUM0 の定数に 10 進数 0 を設定
NUM1	DC	1	定数名NUM1 の定数に 10 進数 1 を設定
NUM2	DC	8	定数名NUM2 の定数に 10 進数 8 を設定
ANS	DS	1	変数名ANS の変数に 1 語 (16 ビット) を確保
	END		プログラム終了

より詳しく調べるために、ループ (繰り返し) ごとにレジスタの値の変化を追ってみる。

ループ回数	GR0	GR1	GR2	GR3
初期値	1	1	1	0
1	2	2	1	1
2	3	3	2	2
3	5	5	3	3
4	8	8	5	4
5	13	13	8	5
6	21	21	13	6
7	34	34	21	7
8	55	55	34	8

以上より、このプログラムは、

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ \cdots$$

のようにフィボナッチ数列を求めている。以上より、変数 ANS に格納されている値は 55 である。