第5章 ソフトウェア

1

授業内容

5章 ソフトウェア

- 5.1 機会命令とアセンブラ
- 5.2 アセンブラ言語とコンパイラ言語
- 5.3 プログラミング手法の分類
- 5.4 コンパイラの仕組み

2

1 機械命令とアセンブラ

1-1 アセンブラ言語

1-2 プログラムの書き方

1-3 レジスタに値を設定する方法

1-4 逐次処理（順次処理）

3

1-1 アセンブラ言語

■ プログラム言語

プログラムとは，どのようなものなのだろうか？

- PCであろうが，携帯電話であろうが，コンピュータ内部で動くプログラムは数値が羅列されたものである．
- 数値の羅列で書かれたプログラムは，コンピュータ（CPU）が唯一理解できるもので，機械語という．

- 高水準言語
  - 人がいかに効率的にプログラムを作れるかということを目的に考え出された言語である．例えば，C, Java, COBOLなど．
- 機械語（machine language：マシン語）
  - デジタル回路・コンピュータが実際に実行している２進数の動作命令であり，一つの動作に対して一つのコードが存在する．
  - 機械の動作を直接指示するため，細かい制御が可能であり，機械の機能を最大限に利用できる．

4

1-1 アセンブラ言語

◆ 高水準言語
- Z = X + Y

コンパイル，コンパイラ，インタープリタ
(compile)　(compiler)　(interpreter)

◆ アセンブリ言語
- LD Rx,0x2000
- LD Ry,0x2001
- ADD Rx, Ry, Rz
- ST Rz, 0x2002

アセンブル，アセンブラー
(assemble)　(assembler)

◆ 機械語
- I/O Rx Ry Rz
- 1
- 1

5

5

1-1 アセンブラ言語

【アセンブラ(assembler)】

- 機械語に翻訳する処理系をアセンブラ(assembler)と呼び，アセンブラで機械語に翻訳することをアセンブル(assemble)と言う．
- 機械語のコードを一対一対応で，自然言語の単語に翻訳したもの．機械語に比べて覚えやすく書き間違いも少ないし，読んで理解しやすい．

【コンパイラ(compiler)】

- 人間が理解しやすい構文を持った高級言語(C，Javaなど)をマシン固有の機械語に翻訳する処理系（ソフトウェア）をコンパイラと言う．
- コンパイラで機械語に翻訳する事をコンパイル(compile)と呼び，コンパイラによって生成された機械語をバイナリ（実行ファイル）と言う．

アセンブラ(assembler)

```
      LAD   GR1,0
      LAD   GR2,0
LOOP  ADDA  GR2,=1
      ADDA  GR2,=1
      CPA   GR2,=10
      JNZ   LOOP
```

機械語(machine language)

```
1210  0000
1220  0000
2020  0001
2412
4020  000A
6200  FFF2
```

6

6

1-1 アセンブラ言語

■ アセンブラ(assembler)言語
- 基本的に機械語と1対1に対応していて，機械語の数字の羅列による命令（数値）を，人に理解しやすい命令（単語など）に置き換えただけの言語である．
- アセンブル(assemble)という作業を行うことで，機械語に変換される．
- 高水準言語でのコンパイル(compile)作業に比べれば，アセンブルの作業は非常に簡単な処理である．
- アセンブラ(assembler)というソフトの開発は容易であるので，JavaやCといった高水準言語が利用できないコンピュータでも，アセンブラ言語を利用する環境は，すぐに用意できる．

7

7

1-1 アセンブラ言語

■ アセンブラ(assembler)言語
- 数値の羅列であるマシン語でプログラムを記述するの極めて困難である。
- アセンブラとマシン語は1対1に対応しているので、マシン語の数値が表す命令にニックネーム（英語を短縮したようなもの）を付け、ニックネームを使ってプログラミングする。
- ニックネームのことを「ニーモニック」と呼び、 CPUのハードウエア的な動作を表すものである。
- アセンブリ言語で記述されたプログラムは、ニーモニックを書き並べたものである。

8

8

1-1 アセンブラ言語

■ アセンブラ言語の特徴

1. メーカーA製，メーカーB製，メーカーC製のコンピュータ（CPU）があると，そのCPUごとに異なる機械語がある．
   - 世の中にアセンブラ言語が山ほどあることになるが，どのアセンブラ言語も基本的な考え方は同じである．
   - 本書では，情報処理技術者試験で使われているCASLⅡというアセンブラ言語を学ぶ．

2. 1つのアセンブラ言語の命令をすべて習得できたということは，それに対応したCPUの命令をすべて習得したことになる．
   - CPUの機能をすべて習得できたことになる．
   - CPUの能力を最大限に引き出すことも可能である．

9

9

1-1 アセンブラ言語

■ COMET-IIとは？
- 産業省基本情報処理技術者試験で出題されるコンピュータ現実には存在しない，試験のための仮想コンピュータである．

✓ Macで動作するフリーウェアーのシミュレーターをダウンロードし，各自時のノートPCにインストールしましょう．

✓ 実際授業の中でも，試験問題やプログラムの検証には，これらのシミュレーターを用います．

■ WCASL-II
- URL
  - http://www.ics.teikyo-u.ac.jp/wcasl2/

- 非常に優れたマニュアル等も付属しており，自習が可能になっている．
  - 「WCASL-IIのUsage」を読む．
  - 「WCASL-IIのTutorial」に従いプログラムを作成し，実行する．

10

10

1-2 プログラムの書き方(1)

プログラム PRG0101

プログラムの書き方

```
1:;プログラム0101
2:PRG0101  START
3:         RPUSH
4:         NOP
5:         RPOP
6:         RET
7:         END
```

実行結果

```
GR0  GR1  GR2  GR3  GR4  GR5  GR6  GR7
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

典型的な形

1. 行番号は不要
2. コメント（;）
3. プログラム開始

11

11

例題

[問] 以下のアセンブラー言語を，命令語の構成表を見ながら，マシン語に変換せよ．

LD　GR1, #0007

LAD　GR2, #0007

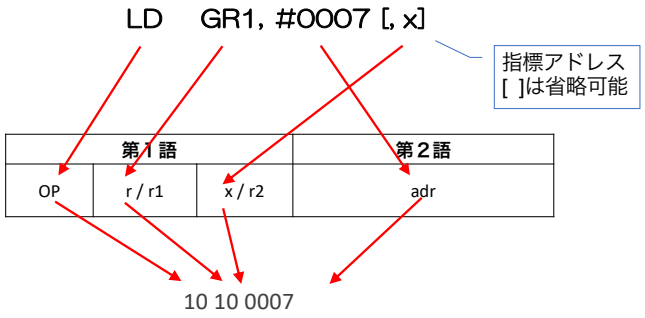
命令語の構成表

https://www.iitec.ipa.go.jp/1_13download/shiken_yougo_ver2_3.pdf

12

12

例題[解答]



13

13

問題[1-1]

以下のアセンブラー言語を，マシン語に変換せよ。

(1) LD GR3, 1, GR2

(2) CPA GR3, GR4

14

14

1-2 プログラムの書き方(2)

- Sample Program

1カラム目 8カラム目 13カラム目

```
;プログラム0101
SAMPLE  START
        LD      GR1,#0027
        ADDA    GR1,#0028
        ST      GR1,#0029
        RET
        DC      1
        DC      2
        DS      1
        END
```

CASLにおける典型的な形（ひな型）

- 注釈欄(Comment Field)：
  ステートメント中にセミコロン（；）があると，その以降，行の終わりまで注釈として扱う．
- ラベル欄(Label Field)：
  ラベルは６文字以内で，先頭の文字は英大文字で始まる．
- 命令コード欄(Instruction Field or Operation Field)：
  各命令の「書き方」の最初に英大文字で書かれている．各命令を示すコード（これを命令コード，またはオペコードという）の部分である．
- オペランド欄(Operand Field)：
  命令の動作を指定する際に使用し，７２文字目までに記述する．

15

15

1-2 プログラムの書き方(3)

- プログラムの開始
  - ラベルとは，
    - 行の先頭(1文字目)から書かれ，その行に名前を付けるためのものである．
    - プログラムの名前を表したり，他の行から参照するために使われる．
    - 1文字以上，8文字以下であり，1文字目はアルファベットから始める．
  - STARTは，
    - プログラムの先頭であることを宣言する命令である．
    - プログラムを実行するとき，この行から実行を開始するということを宣言するものである．

16

16

1-2 プログラムの書き方(4)

- プログラムの終了
  - RET命令（リターン命令）：
    - プログラムの動作を終了させるための命令である.
  - END命令：
    - プログラムの記述の終了を示す命令である.

- アセンブラ言語のプログラムのひな型

```
プログラム名  START
        実質的なプログラムを記入
        RET
        DS命令，DC命令
        END
```


17

1-3 レジスタに値を設定する方法

➢汎用レジスタと呼ばれる記憶装置に，値を設定する

```
1:;Program0101
2:PRG0101 START
3:          LAD     GR7,#1234
4:          RET
5:          END
```

	GR0	GR1	GR2	GR3	GR4	GR5	GR6	GR7
3行目実行以前	0000	0000	0000	0000	0000	0000	0000	0000
3行目実行以後	0000	0000	0000	0000	0000	0000	0000	1234

18

1-3 レジスタに値を設定する方法

■ プログラムの概要

【1行目】　;Program0101
- “；” の文字で始まっているので，コメントの行である.

【2行目】　PRG0101 START
- 1文字目から書かれているラベルPRG0102はプログラム名である.
- この行で名前PRG0102のプログラムを開始することを示している.

【4, 5行目】
- 4行目のRET命令は，このプログラムの実行を終了するための命令である
- 5行目のEND命令で，記述しているプログラムの最終行であることを示す

19

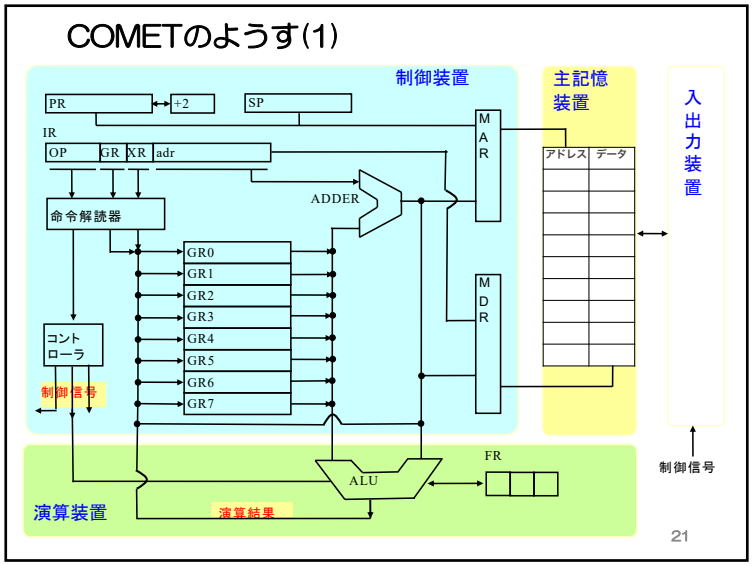
1-3 レジスタに値を設定する方法

■ LAD命令

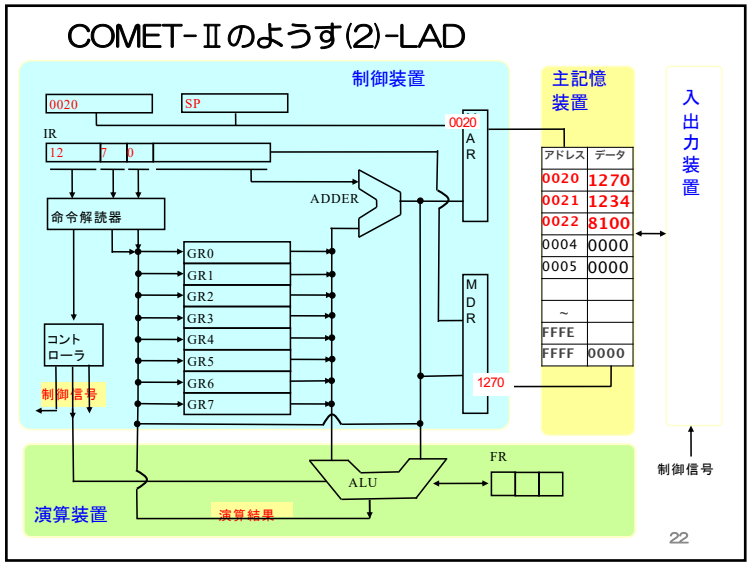
【3行目】　LAD　GR7,#1234
- “LAD”は，命令コードであり，“GR7,#1234”はオペランドと呼ばれる部分である.
- オペランドは，命令で利用するデータに関する情報が書かれている箇所であり，命令コードごとにその内容や書き方（書式）が変わる.
- “GR7”は，汎用レジスタ(General Register)の7番を意味する.（予約語）
- “#1234”は，数値の頭に“#”を書くと16進数で，1234という数値を表す.

【意味】
- 値1234_{16}を汎用レジスタGR7に設定する

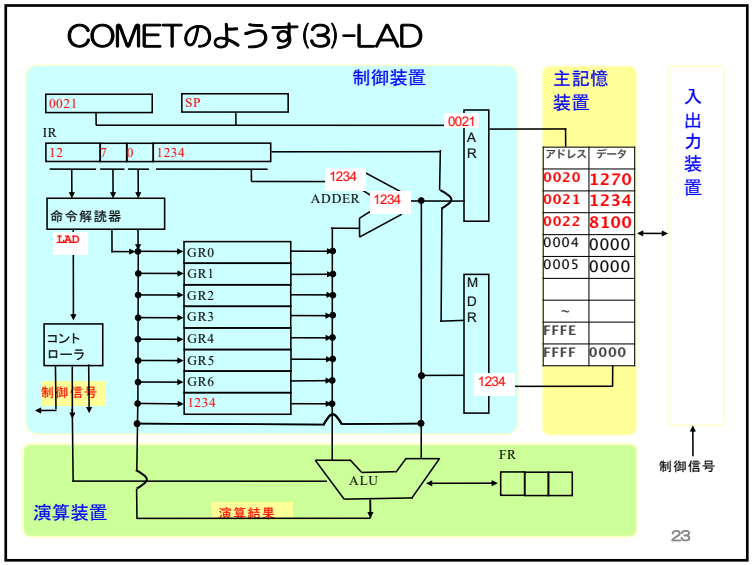
20



21



22



23

1-3 レジスタに値を設定する方法

■ 定義

命令	書き方		命令の説明
	命令コード	オペランド	
ロードアドレス Load Address	LAD	r,adr	汎用レジスタrに値adrを設定する r ← adr rはGR0～GR7

■比較

– **LAD GR7,1234** ；10進数の場合
– **LAD GR7,#1234** ；16進数の場合

■注意

– LAD命令には，オペランドが必要でカンマで区切って書く必要がある．
– オペランドのなかに間隔文字（空白）を入れてはいけない．

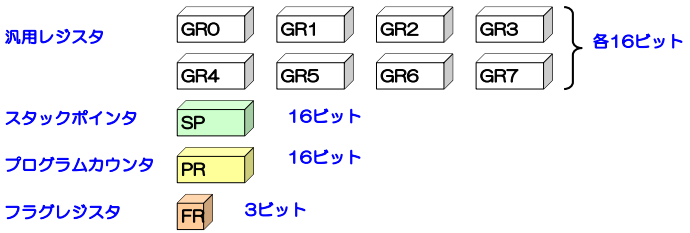
24

24

1-3 レジスタに値を設定する方法

■ レジスタとは

- コンピュータのCPUにある最も重要な記憶装置が，レジスタと呼ばれる記憶装置である．
- 本書で扱うような最もシンプルな構成のCPU（COMETⅡ）から，最近のPCなどの高性能のコンピュータまで，ほぼすべてのコンピュータに備わっている．



25

1-3 レジスタに値を設定する方法

■ レジスタの種類と数は，CPUあるいはコンピュータシステムによってさまざまであり，また呼び方もさまざまであるが，ほとんどのコンピュータが，この4種類の機能のレジスタを持っている．

- 汎用レジスタ

  COMETⅡには，汎用レジスタと呼ばれるGR0～GR7の8個のレジスタがある．これらの汎用レジスタは，アセンブラ言語のプログラムで自由に使用できるレジスタである．この汎用レジスタには，記憶装置としての機能と，演算装置としての機能がある．

- スタックポインタ（SP）

  サブルーチンというプログラムの仕組みを容易に実現するための役割を担うレジスタである．アセンブラ言語CASLⅡでは，スタックポインタ（SP）を，汎用レジスタ（GR0～GR7）のように，命令によって直接操作することはできない仕様になっている．したがって，すでに出てきたRPUSH命令，RPOP命令やPUSH命令，POP命令などによってスタックポインタを間接的に使用することになる．

- プログラムレジスタ（PR）

  いまどこの命令を実行しているかというプログラムの動作を管理するためのレジスタである．このレジスタは，ノイマン型コンピュータと呼ばれる現在のコンピュータには，なくてはならない存在である．プログラムを作成するときに，このレジスタの詳細な値の変化を直接意識する必要はない．しかし，プログラムの処理の流れを制御するとき，ラベルによって，間接的にPRを操作することになる．

- フラグレジスタ（FR）

  CPUにより行われた演算結果の状態を設定するレジスタである．例えば，演算結果が正か負かゼロかといった状態や，2つの値を比較するといった演算でどちらの値が大きいかといった状態を記録するのが，このフラグレジスタである．

26

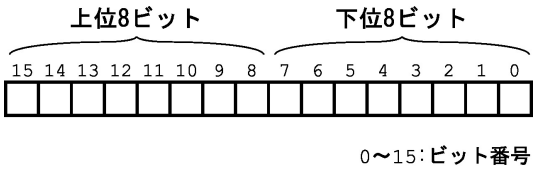
1-3 レジスタに値を設定する方法（まとめ-1）

■ 1ワード（語，Word）：情報の基本サイズ

■ COMETⅡでは 16ビット(bit)

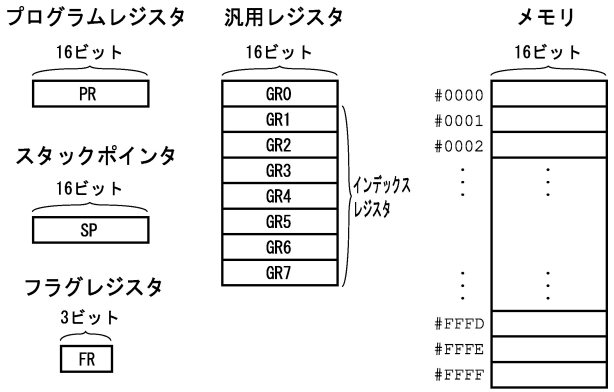
■ 数値の場合

- 2の補数表現
- 最上位ビットは正負符号



27

1-3 レジスタに値を設定する方法（まとめ-2）



28

問題[1-3]

汎用レジスタGR1～GR3に以下の値を順に設定するプログラムを作成しなさい．

GR1 <-- 0010
GR2 <-- #0010
GR3 <-- #000A

29

29

1-4 逐次処理（順次処理）

□プログラムが実行される順序を理解するために，LAD命令をいくつも書き並べたプログラムである．

```
1:;プログラム0103
2:PRG0103 START          ;プログラムを開始
3:        RPUSH          ;決まり文句
4:        LAD GR0,#1234  ;GR0に16進数1234を格納
5:        LAD GR1,#5678  ;GR1に16進数5678を格納
6:        LAD GR2,#9ABC  ;GR2に16進数9ABCを格納
7:        LAD GR3,#DEF0  ;GR3に16進数DEF0を格納
8:        LAD GR4,1234   ;GR4に10進数1234を格納
9:        LAD GR5,5678   ;GR5に10進数5678を格納
10:       LAD GR6,9876   ;GR6に10進数9876を格納
11:       LAD GR7,5432   ;GR7に10進数5432を格納
12:       LAD GR1,#2468  ;GR1に16進数2468を格納
13:       RPOP           ;RPUSHとペアで決まり文句
14:       RET            ;プログラム終了
15:       END            ;プログラム一番最後の決まり文句
```

30

30

1-4 逐次処理（順次処理）

	GR0	GR1	GR2	GR3	GR4	GR5	GR6	GR7
(a) 4行目実行前	0000	0000	0000	0000	0000	0000	0000	0000
(b) 4行目実行後	1234	0000	0000	0000	0000	0000	0000	0000
(c) 5行目実行後	1234	5678	0000	0000	0000	0000	0000	0000
(d) 6行目実行後	1234	5678	9ABC	0000	0000	0000	0000	0000
(e) 7行目実行後	1234	5678	9ABC	DEF0	0000	0000	0000	0000
(f) 8行目実行後	1234	5678	9ABC	DEF0	04D2	0000	0000	0000
(g) 9行目実行後	1234	5678	9ABC	DEF0	04D2	162E	0000	0000
(h) 10行目実行後	1234	5678	9ABC	DEF0	04D2	162E	2694	0000
(i) 11行目実行後	1234	5678	9ABC	DEF0	04D2	162E	2694	1538
(j) 12行目実行後	1234	2468	9ABC	DEF0	04D2	162E	2694	1538

31

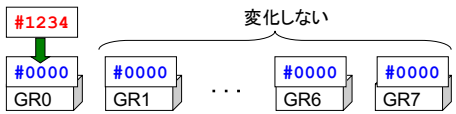
31

1-4 逐次処理（順次処理）

■ プログラム実行の順序

【4行目】 LAD GR0,#1234 ;GR0に16進数1234を格納

- 命令の仕様：GR0 ← 1234_{16}
  （←は，値を汎用レジスタに設定する意味を表している）
- GR0に16進数の1234_{16}を設定という動作である．［実行のようす（a）］
- 4行目実行前は，すべての汎用レジスタに0000_{16}の値が設定されている．［実行のようす（b）］
- 4行目のLAD命令を実行すると，汎用レジスタGR0の値だけが1234_{16}に変わり，GR1～GR7の値は0000_{16}のままで変化しない．



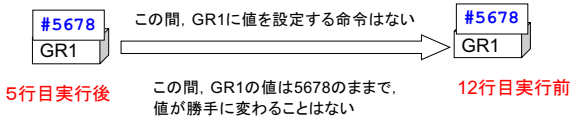
32

32

1-4 逐次処理（順次処理）

■まとめ

- プログラムに書いた命令は上から下に順に実行されるということである
  - 現在のほとんどすべてのコンピュータの基本となっている動作である
- 命令実行とは関係のない汎用レジスタの値は，前の状態を保持している



5行目実行後

この間，GR1の値は5678のままで，
値が勝手に変わることはない

12行目実行前

33

33

問題[1-4]

汎用レジスタGR4の値が，以下のように順に変化するプログラムを作成しなさい．

AAAA → BBBB → CCCC → DDDD

34

34