コンピュータ概論B - ソフトウェアを中心に -

#7 プログラムとアルゴリズム

Yutaka Yasuda

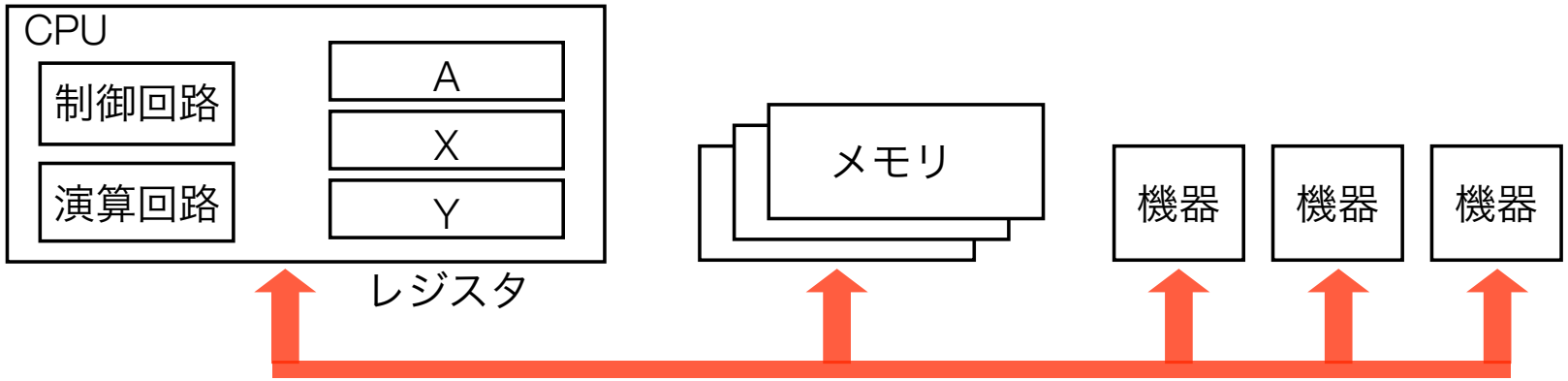
プログラミング

- 教科書 p.60〜
- ソフトウェア、プログラム、アルゴリズムの関係
- プログラミング言語

機械語の例

- CPUに対する命令
  （CPU の機能を指定して実行）
- CPU の構造
  レジスタの存在
  メモリとレジスタを利用して計算結果を保持しながら連続処理
- 命令はメモリに連続的に配置

- CPU にできること
  レジスタ・メモリ間のデータのやりとり
  レジスタ・レジスタ間またはレジスタ・メモリ間の演算や比較、条件分岐
  機器回路の制御



機械語の例

MOS Technology の 6502 で 12 と 23 を足す例

アセンブリ言語

```
LDA $1002
CLR
ADC $1003
STA $1002
```

```
AD 02 10
18
6D 03 10
8D 02 10
```

これが機械語

メモリの中ではこんな感じ

	AD
	02
	10
	18
	6D
	03
	10
	8D
	02
	10
1002	12
1003	23

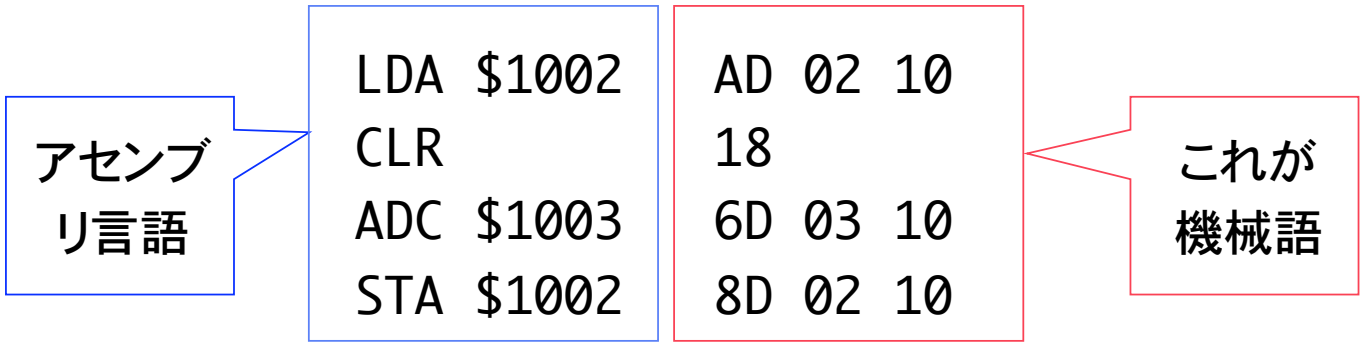
1. 1002番地の値をアキュムレータに読み
2. 桁上がり無しで
3. 1003番地にある値を加え
4. 1002番地に書き込み

アセンブリ言語

- 機械語を読みやすい形式（ニモニック）で表現したもの

  このように表現するだけでかなり読みやすくなる。

- 実際には複雑な機能もあるがここでは紹介しない



マシン語とアセンブリ言語

- マシン語
  - ハードウェアにべったり依存
  - レジスタの名前や本数、CPU 命令（=回路）の番号
  - CPU設計時に言語仕様が決定
- アセンブリ言語
  - マシン語とほぼ一対一対応
  - レジスタ本数などハードウェア依存の性質が残る
- 低級言語
  - 実際にプログラミングする際には、どのCPU上で実行されるか意識してプログラミングする必要あり

アセンブリ言語の例

PowerPC 750

```
lwz r9,$1002    1002 番地の値(word)を Gr9 に
lwz r0,$1006    1006 番地の値(word)を Gr0 に
add r0,r9,r0     r0+r9 を r0 に
stw r0,$1002    結果 r0 を 1002 番地に
```

MOS Technology 6502

```
LDA $1002       AD 02 10
CLR             18
ADC $1003       6D 03 10
STA $1002       8D 02 10
```

高級言語の例

- 要求

  ハードウェアに依存しないプログラミング

  長持ちして欲しい

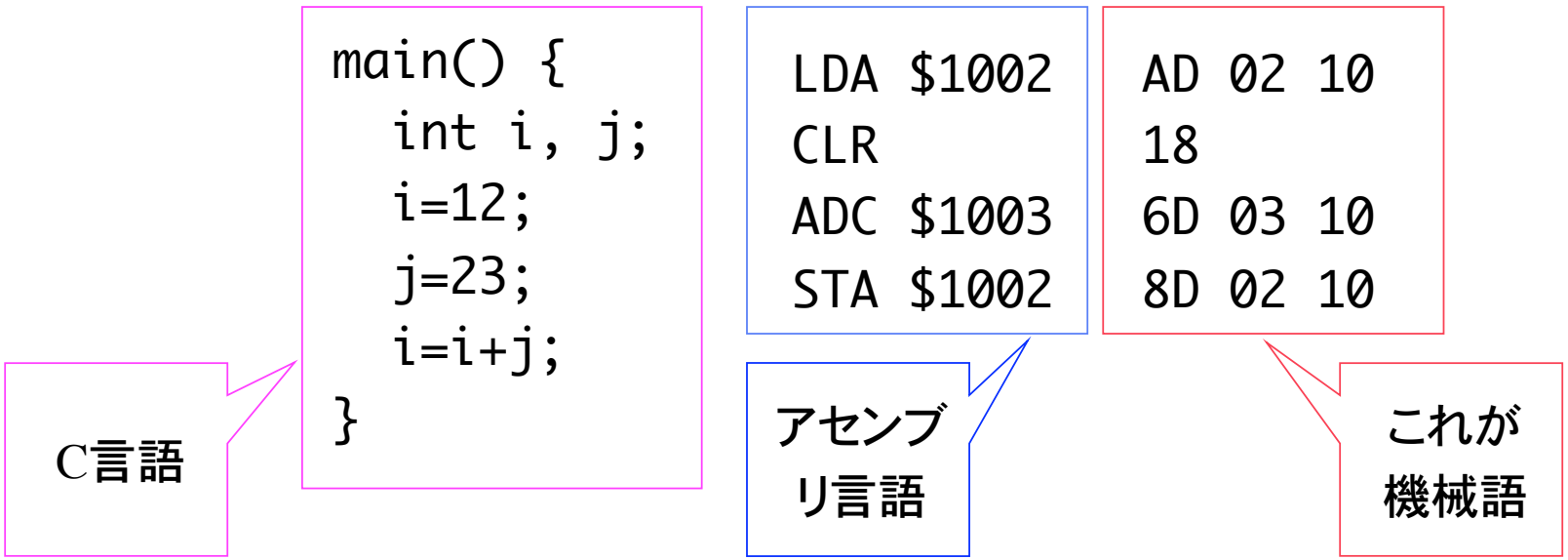
  いろんなシステムで動いて欲しい

- 結果

  高水準言語で書いて、低水準言語（または機械語）に変換すれば良い

高級言語の例（C言語）

- メモリの構造やレジスタの名前を意識しないプログラムが書けている
- 抽象度が高くなっている、と表現する



抽象度

- 二つの整数を加算する場合
- 機械語

  一時使用するレジスタ番号、メモリアドレスを指定

- アセンブリ言語

  レジスタ番号などはまだ残る

- 高級言語

  ハードウェアの中身から独立している

  抽象度が高まった、移植性が高まった

47 F0 E0 57
F0 E4 E7 F0
E8

LD G1, F0E0
ADD G1, F0E4
ST G1, F0E8

a=b+c

自然言語と人工言語

- 自然言語
  人間が日常生活で用いている言語 (日本語、英語、etc..)
- 人工言語
  人間が意図的に作り出した言語
  エスペラント (1887年 L.L. Zamenhof , ポーランド)
  全てのプログラミング言語
- プログラミング言語の特徴
  決定的な動作のための明確な手順指示書
  一点一字の間違いも許されない
  限定的な語彙（C言語では予約語は 32）
  簡単な文法

低級言語

- 例えばアセンブリ言語
  - 自然言語に近いほど「高水準」と考える
  - 歴史的には低水準から高水準へと進化
- CPU 依存（CPU の機能を直接指定＝移植性がない）
- 機械語とほぼ一対一
- ニーモニックコード (mnemonic : 記憶を助ける)
- 可読性が悪く、保守性が悪い
- 作成が難しく、生産性が低い
- アセンブラ(Assembler)と呼ばれるソフトウェアを用いて機械語に変換して実行

高級言語

- 例えば C 言語
- CPU 依存性なし (CPU 命令は直接指定できない)
- メーカー、OS が変わっても再利用できる(移植性が高い)
- 抽象度が高く、アセンブリ言語よりは自然言語に近い（プログラマにやさしい）語彙と文法
- コンパイラ(Compiler)と呼ばれるソフトを利用してアセンブリ言語または機械語に変換
- 可読性が高く、保守しやすい
- プログラミングが容易で生産性が高い

コンパイラ

C 言語

```
main()
{
  int i,j;
  i=1954;
  j=1959;
  i=i+j;
}
```

PowerPC 750 アセンブリ言語

```
.text
        .align 2
        .globl _main
_main:
        stmw r30,-8(r1)
        stwu r1,-64(r1)
        mr r30,r1
        li r0,1954          Gr0に 1954
        stw r0,32(r30)      Gr0 を Gr30 から 32 バイト先(i)に格納
        li r0,1959          Gr0に 1959
        stw r0,36(r30)      こんどは 36 バイト先に格納
        lwz r9,32(r30)      i を Gr9 に
        lwz r0,36(r30)      j を Gr0 に
        add r0,r9,r0        Gr0+Gr9 を Gr0 に
        stw r0,32(r30)      結果 Gr0 を i に
        mr r3,r0
        lwz r1,0(r1)
        lmw r30,-8(r1)
        blr
```

アルゴリズム

- 教科書 p.63〜
- アルゴリズムをプログラミング言語で書き下すことがプログラミングである。