東邦大学

生命（いのち）の科学で未来をつなぐ

COMET IIのプログラミング

ここでは
機械語レベル プログラミング
を学びます

ここでは
機械命令レベル プログラミング
を学びます

機械命令の形式は学びましたね
機械命令を並べたプログラムを作ります

東邦大学

その前に

東邦大学

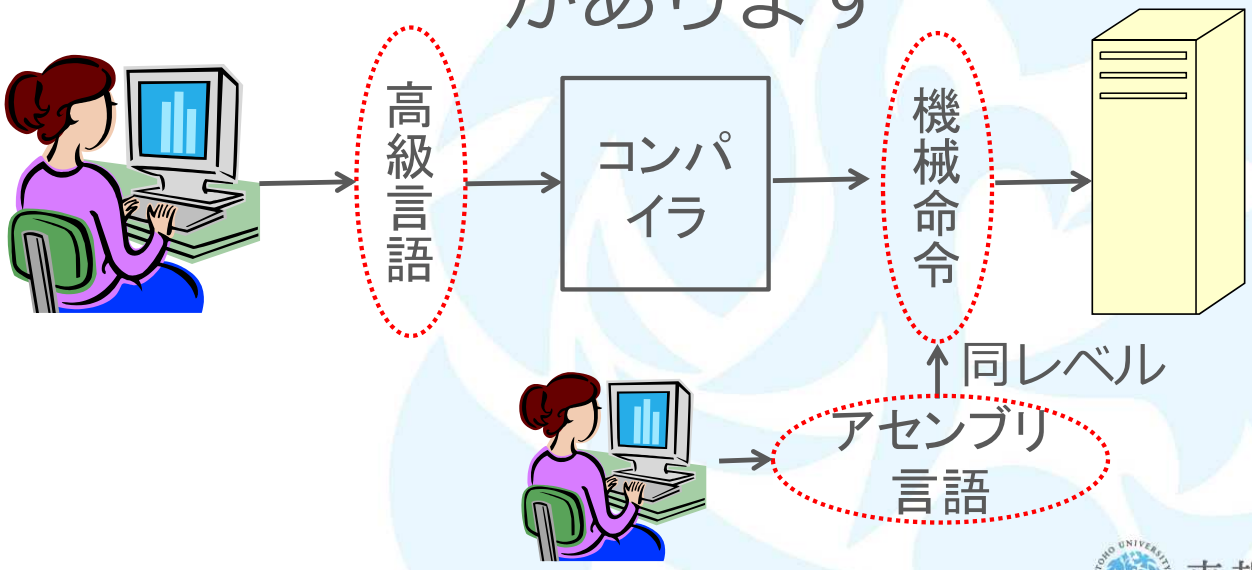
プログラミング言語について

プログラミング言語について

高級言語（JavaとかCとか）と
機械命令レベルの言語（アセンブリ言語）
があります

プログラミング言語について

高級言語（JavaとかCとか）と
機械命令レベルの言語（アセンブリ言語）
があります



プログラミング言語について

高級言語（JavaとかCとか）と
機械命令レベルの言語（アセンブリ言語）
があります

高級言語 ⇒ 人間に近い ⇒ 書きやすい

アセンブリ言語 ⇒ 機械に近い
⇒ プログラムの効率が良い

と言われる

ここで挑戦するアセンブリ言語は

- 効率よりは、機械の動作を理解することが目的
- あまり細かいことは気にしない

ここで挑戦するアセンブリ言語は

- 効率よりは、機械の動作を理解することが目的
- あまり細かいことは気にしない

- でも、一応はアセンブリ言語の形になったプログラムを書きましょう

もう一つ、お断り

ここで挑戦するアセンブリ言語は

- 機械が実在しない（仮想的）なものを使います

  情報処理技術者試験で使われる
  仮想CPU COMET-II のための
  アセンブリ言語 CASL-II を使う

http://www.jitec.jp/1_13download/shiken_yougo_ver2_0.pdf

ここで挑戦するアセンブリ言語は

- 機械が実在しない（仮想的）なものを使います

  情報処理技術者試験で使われる
  仮想CPU COMET-II のための
  アセンブリ言語 CASL-II を使う

たとえばIntelのPC用CPU (Core-3/5/7iなど)を使ってもよいのだが、命令が複雑で、最初の学習には向かないだろう

ようやく本論
アセンブリ言語 CASL-II

まず書き方の規則

まず書き方の規則

- １行に１命令
- [ラベル:] OPコード,オペランド(複数)
- オペランドは命令によっていろいろ

まず書き方の規則

- １行に１命令
- [ラベル]: OPコード,オペランド(複数)
- オペランドは命令によっていろいろ

LABEL1:　ADDA GR3, 2537

まず書き方の規則

- １行に１命令
- [ラベル]: OPコード,オペランド(複数)
- オペランドは命令によっていろいろ

LABEL1:　ADDA GR3, 2537

ラベル　OPコード オペランド

まず書き方の規則

- １行に１命令
- [ラベル]: OPコード,オペランド(複数)
- オペランドは命令によっていろいろ

LABEL1:　ADDA GR3, 2537

ラベル　OPコード　オペランド

命令（何をするか）を指定する

ADDA＝足し算をする

まず書き方の規則

- １行に１命令
- [ラベル]: OPコード,オペランド(複数)
- オペランドは命令によっていろいろ

LABEL1:　ADDA GR3, 2537

ラベル　OPコード　オペランド

操作の対象を指定する

GR3に2537番地の内容を足す

まず書き方の規則

- １行に１命令
- [ラベル]: OPコード,オペランド(複数)
- オペランドは命令によっていろいろ

LABEL1:　ADDA GR3, 2537

ラベル　OPコード オペランド

まず書き方の規則

- １行に１命令
- [ラベル]: OPコード,オペランド(複数)
- オペランドは命令によっていろいろ

LABEL1:　ADDA GR3, 2537

ラベル　OPコード オペランド

この命令の置いてある場所を示す

LABEL1で示される場所に置いてある

命令が並ぶと

```
LD   GR3, 201
ADDA GR3, 202
ST   GR3, 203
```

メモリ201番地の内容を
汎用レジスタ3へロード(コピー)

メモリ202番地の内容と汎用レジスタ3を足してレジスタ3へ格納

汎用レジスタ3の内容を
メモリ203番地へストア(コピー)

命令が並ぶと

①`LD   GR3, 201`

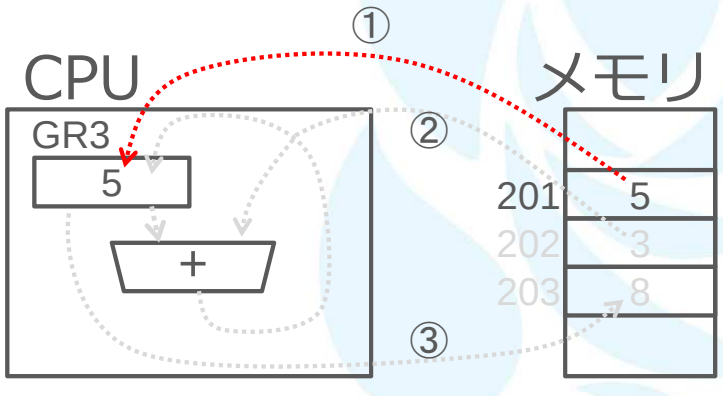
メモリ201番地の内容を
汎用レジスタ3へロード(コピー)

②`ADDA GR3, 202`

メモリ202番地の内容と汎用レジスタ3を足してレジスタ3へ格納

③`ST   GR3, 203`

汎用レジスタ3の内容を
メモリ203番地へストア(コピー)



命令が並ぶと

①LD　GR3, 201

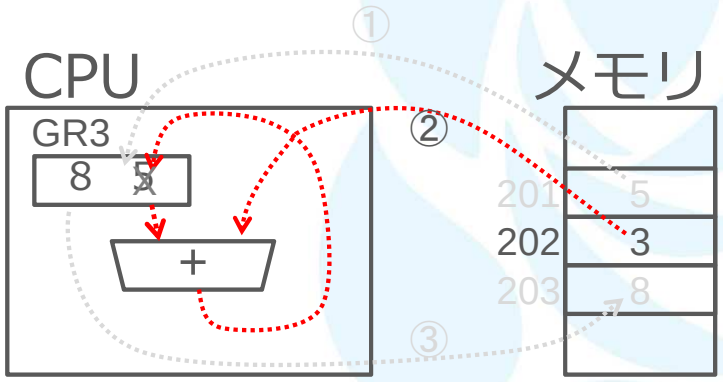
②ADDA GR3, 202

③ST　GR3, 203

メモリ201番地の内容を
汎用レジスタ3へロード(コピー)

メモリ202番地の内容と汎用レジスタ3を足してレジスタ3へ格納

汎用レジスタ3の内容を
メモリ203番地へストア(コピー)



命令が並ぶと

①LD　GR3, 201

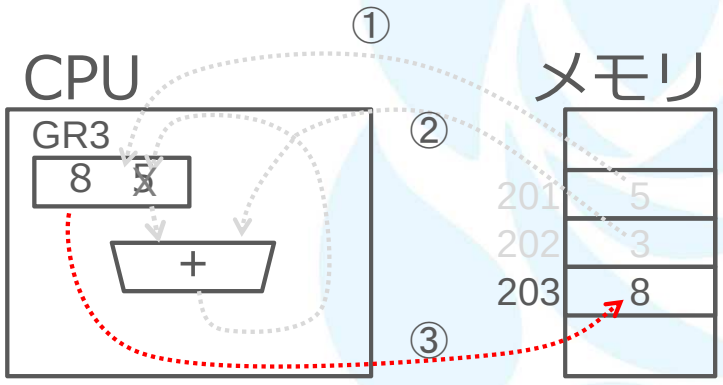
②ADDA GR3, 202

③ST　GR3, 203

メモリ201番地の内容を
汎用レジスタ3へロード(コピー)

メモリ202番地の内容と汎用レジスタ3を足してレジスタ3へ格納

汎用レジスタ3の内容を
メモリ203番地へストア(コピー)



命令が並ぶと

① LD　GR3, 201

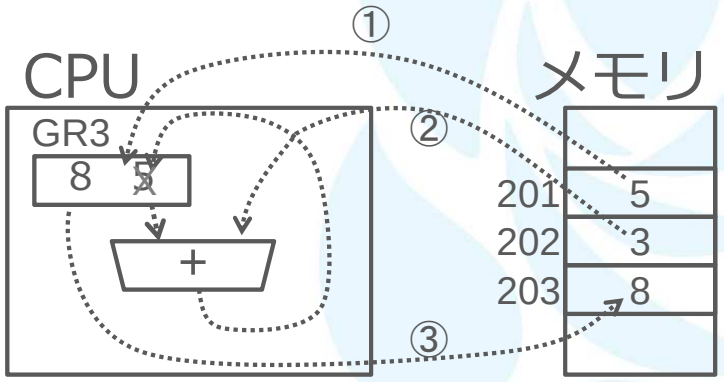
② ADDA GR3, 202

③ ST　GR3, 203

メモリ201番地の内容を
汎用レジスタ3へロード(コピー)

メモリ202番地の内容と汎用レジスタ3を足してレジスタ3へ格納

汎用レジスタ3の内容を
メモリ203番地へストア(コピー)



このように
命令を１行ずつ
順番に実行する

ここまでまとめると

- １行に１命令
- [ラベル:] OPコード,オペランド の形
- OPコード＝何をする命令か(命令種類)
  オペランド＝命令の操作対象
- 上から順に１行ずつ処理が進む(実行)

もう少し先へ行こう

⇒ オペランドの書き方

オペランド

LD GR3,201

「メモリ上の201番地の内容」

オペランド

LD GR3,201

「メモリ上の201番地の内容」

⇩

プログラマのイメージは、x = x + y

（名前の付いた）変数

オペランド

LD GR3,201

「メモリ上の201番地の内容」

⇩

プログラマのイメージは、x = x + y

（名前の付いた）変数

番地でなく
名前で呼びたい

⇩

「メモリ上の変数 y 」

オペランド

LD GR3,201

「メモリ上の201番地の内容」

⇩

プログラマのイメージは、x = x + y

（名前の付いた）変数

番地でなく
名前で呼びたい

⇩

「メモリ上の変数 y 」

⇩

LD GR3, y　と書く。但しyは予めメモリ上に取った変数の名前

オペランド

LD GR3,201

「メモリ上の201番地の内容」

⇩

プログラマのイメージは、x = x + y

（名前の付いた）変数

番地でなく
名前で呼びたい

⇩

「メモリ上の変数 y 」

但しこの授業では
厳密にはしない

厳密には他の本を見てください

⇩

LD GR3, y　と書く。但しyは予めメモリ上に取った変数の名前

もう少し先へ行こう

⇒ オペランドの書き方

オペランド

LD GR3,201

「汎用レジスタ３」

オペランド

LD GR3,201
「汎用レジスタ３」

COMET IIでは、汎用レジスタは
GR0, GR1, … , GR7　の８つ

オペランド

LD GR3,201
「汎用レジスタ３」

COMET IIでは、汎用レジスタは
GR0, GR1, … , GR7　の８つ

どれをどう使ってもよい
（プログラマの勝手）が
GR0だけは指標レジスタとして使えない

これも、この授業では厳密には制限しない

脱線かな？　いろいろな命令

⇒ COMET II の命令一覧

COMET IIの命令の資料

教科書 28ページ
資料：
情報処理技術者試験
「試験で使用する情報技術に関する用語・プログラム言語など」ver 2.2　　の　別紙１　の　3～9ページ

https://www.jitec.ipa.go.jp/1_13download/shiken_yougo_ver2_2.pdf

命令の種類と動作は、3～4ページの表(5ページは使わない)を参照すること

COMET IIの命令の資料

命　令	書き方		命令の説明	FRの設定
	命令コード	オペランド		
(2) 算術，論理演算命令				
算術加算 ADD Arithmetic	ADDA	r1,r2	r1 ← (r1) + (r2)	○
		r,adr [,x]	r ← (r) + (実効アドレス)	
論理加算 ADD Logical	ADDL	r1,r2	r1 ← (r1) $+_L$ (r2)	
		r,adr [,x]	r ← (r) $+_L$ (実効アドレス)	
算術減算 SUBtract Arithmetic	SUBA	r1,r2	r1 ← (r1) − (r2)	
		r,adr [,x]	r ← (r) − (実効アドレス)	
論理減算 SUBtract Logical	SUBL	r1,r2	r1 ← (r1) $-_L$ (r2)	
		r,adr [,x]	r ← (r) $-_L$ (実効アドレス)	

COMET IIの命令の資料

命　令	書き方		命令の説明	FRの設定
	命令コード	オペランド		
(2) 算術，論理演算命令				
算術加算 ADD Arithmetic	ADDA	r1,r2	r1 ← (r1) + (r2)	○
		r,adr [,x]	r ← (r) + (実効アドレス)	
論理加算 ADD Logical	ADDL	r1,r2	r1 ← (r1) $+_L$ (r2)	
		r,adr [,x]	r ← (r) $+_L$ (実効アドレス)	
算術減算 SUBtract Arithmetic	SUBA	r1,r2	r1 ← (r1) − (r2)	
		r,adr [,x]	r ← (r) − (実効アドレス)	
論理減算 SUBtract Logical	SUBL	r1,r2	r1 ← (r1) $-_L$ (r2)	
		r,adr [,x]	r ← (r) $-_L$ (実効アドレス)	

COMET IIの命令の資料

命　令	書き方		命令の説明	FRの設定
	命令コード	オペランド		
(2) 算術，論理演算命令				
算術加算 ADD Arithmetic	ADDA	r1,r2	r1 ← (r1) + (r2)	○
		r,adr [,x]	r ← (r) + (実効アドレス)	
論理加算 ADD Logical	ADDL	r1,r2	r1 ← (r1) $+_L$ (r2)	
		r,adr [,x]	r ← (r) $+_L$ (実効アドレス)	
算術減算 SUBtract Arithmetic	SUBA	r1,r2	r1 ← (r1) − (r2)	
		r,adr [,x]	r ← (r) − (実効アドレス)	
論理減算 SUBtract Logical	SUBL	r1,r2	r1 ← (r1) $-_L$ (r2)	
		r,adr [,x]	r ← (r) $-_L$ (実効アドレス)	

このぐらいにして、
具体的なプログラミングに
移ろう