
プロセッサ

コンピュータ基礎 (7)

菊池浩明

講義概要

■ 教科書

- 4章 プロセッサ
- 1. 基本機能
- 2. 構成回路
- 3. コンピュータアーキテクチャー
- 4. 命令の種類と形式
 - » オペランド
 - » 有効アドレス
- 5. 動作の流れ

1. 基本機能

- 構成要素

- 入力部

- 記憶部

- 演算部

- 制御部

- » いわゆるプロセッサ, CPU

- 出力部

2. 構成回路

- 命令実行部

- PC: プログラムカウンタ
- IR: 命令レジスタ (Instruction Register)
- REG: レジスタ群

- 演算実行部

- ALU: 算術演算回路 (Arithmetic and Logic Unit)
- AC: アキュムレータ (Accumulator)
- マイクロプログラム制御

3. コンピューターアーキテクチャー

■ Architecture

- 建築術, 建物, 構造

- ネットワークアーキテクチャー

- CPUアーキテクチャー

 - » 命令セット, メモリアーキテクチャ, バッファメモリ

 - » 仮想記憶

4. 命令 (機械語)

種類	内容	例 (CASL)
データ転送	主記憶とレジスタ間データ転送	LD, LAD, ST
演算命令	算術演算, 論理演算, 比較演算	ADDA, SUBA, AND, OR, XOR, SLA, SRA, CPA
制御, 分岐命令	プログラムの制御	JPL, JMI, JZE, JNZ, JUMP, PUSH, POP CALL, RET
入出力命令	周辺装置との入出力	IN, OUT
特殊命令	割り込み	SVC

CASL II

■ COMET II

- 基本情報技術者試験用仮想機械
- 最低限の機能を持つ簡素なアーキテクチャー
- 16-bit (ワード) 汎用レジスタ: GR0,...,GR7
- 主記憶 65,536ワード (128MB)
- 16-bit プログラムレジスタ: PR (PC)

■ CASL II

- COMET用仮想アセンブラ

CASLシュミレーター

ソースコードをペースト

ステップ
イン

実行

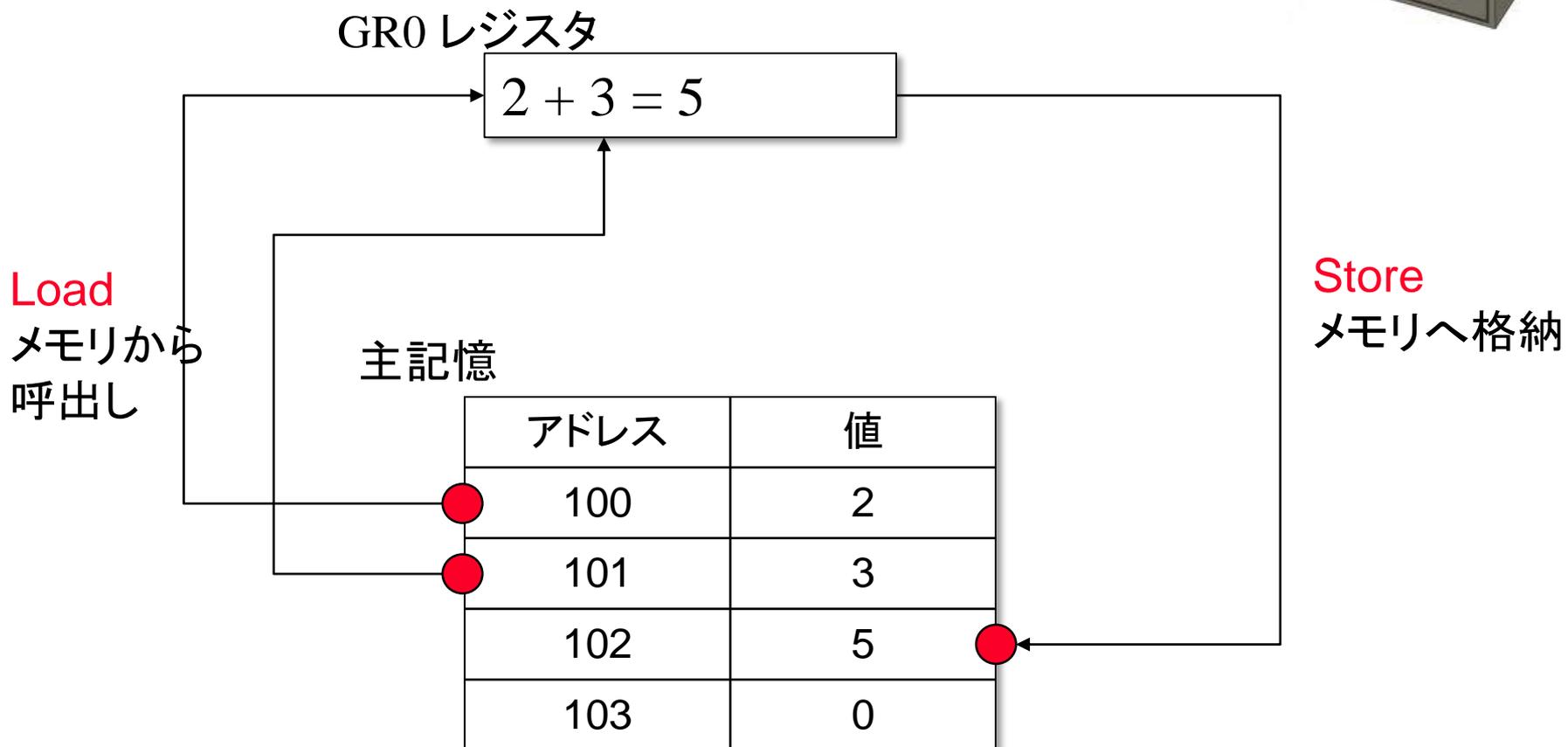
汎用レジ
スタ

メモリの中
身(アドレ
ス指定)

The screenshot displays the CASL simulator interface with several key components:

- Source Code Window:** Shows assembly code with instructions like PRG, BEGIN, LAD, LD, LOOP, NEXT, CPA, JMI, and ST. A callout bubble points to the code with the text "ソースコードをペースト".
- Control Panel:** Includes buttons for step-in, execution, and step-out, along with a "すべてのブレークポイントを削除" (Remove all breakpoints) button and a "中止" (Stop) button.
- Register Window:** Displays the state of registers GR0 through GR7, PR, SP, ZF, SF, and OF. A callout bubble points to this section with the text "汎用レジスタ".
- Memory Window:** Shows memory addresses (e.g., 0010, 0014, 0018, 001C) and their corresponding values. A callout bubble points to this section with the text "メモリの中身(アドレス指定)".
- Console Window:** Displays the execution progress, showing the current instruction address (0004) and the instruction being executed (LD GR0, A). A callout bubble points to the console with the text "プログラムカウンタが表示行".

主記憶とレジスタ



アセンブラと機械語

■ アセンブラ(CASL)

■ 機械語(COMET)

ニモニク(命令)

オペランド

```
PGM START BEGIN
BEGIN LD GR0, A
      LAD GR1, A
      LD GR2, 0, GR1
      LAD GR3, 0, GR1
      RET
A DC #27
B DC #1D
C DS 1
アセンブラ命令 END
```

アセンブル

"バイナリ"

0000	1000 0009
0002	1210 0009
0004	1021 0000
0006	1231 0000
0008	8100
0009	0027
000A	001D
000B	0000

記号アドレス

逆アセンブル

アセンブラ命令

Sample 1

```
PGM   START BEGIN
BEGIN LD   GR0,A
      LAD  GR1,A
      LD   GR2,0,GR1
      LAD  GR3,0,GR1
      RET
A     DC   #27
B     DC   #1D
C     DS   1
      END
```

アセンブラと高級言語

- アセンブラ (CASL)
 - アセンブル言語 → 機械語 (一対一対応)
- コンパイラ (C)
 - 高級言語 → 機械語 (最適化)
- インタプリタ (Perl, PHP)
 - 高級言語 → インタプリタ
 - 高級言語 → 中間言語 → 仮想機械 (Java, Processing)

転送命令

■ レジスタ間転送 **LOAD**命令

LD レジスタ1, レジスタ2

□例) LD GR0, GR1

GR1の内容をGR0にコピー GR0 ← GR1

□GR0,...,7が利用可能

直接レジスタ設定

■ LOAD ADDRESS命令

LAD レジスタ*r*, アドレス*a*

- アドレス*a*(値)をレジスタ*r*に設定する
- 例1) LAD GR0,100 GR0=100
- 例2) LAD GR1, A GR1=0009 (A記号
 アドレスの値)
- 例3) LAD GR2, GR0 GR2=

直接アドレスからロード命令

■ Load命令

LD レジスタr, アドレスa

□アドレスaで指定されたメモリの値をレジスタrにロードする.

□例)

LD GR0,09 ;GR0=27

LD GR1,B ;GR1=

LAD GR2,C ;GR2=

記号	アドレス	値
A	0009	0027
B	000A	001D
C	000B	0000

インデックス(指標)レジスタ参照**

■ アドレス修飾ロード

LD レジスタr, アドレスa, インデックスレジスタx

□アドレスa+インデックスレジスタxで指定されるメモリの値をrにロードする.

□例) GR1=1, GR2=2の時

LD GR0, A, GR1 ; GR0=1D

LD GR3, A, GR2 ; GR3=

LD GR4, B, GR1 ; GR4=

記号	アドレス	値
A	0009	0027
B	000A	001D
C	000B	0000

□注: GR0はインデックスレジスタに利用できない

格納命令

■ メモリへの格納(**STORE**)

ST レジスタr, アドレスa [,x]

□レジスタrの内容をアドレスaで指定されたメモリに格納する.

□例) GR0=2, GR1=0009, GR2=1の時

ST GR0, A ; 9番地を2に

ST GR2, A, GR2

ST GR1, A, GR0

記号	アドレス	値
A	0009	2
B	000A	
C	000B	

□アドレス修飾は全ての命令で利用可能

算術命令

■ 加算 (Add Arithmetic), 減算 (Subtract)

ADDA レジスタr1, レジスタr2

SUBA レジスタr1, レジスタr2

ADDA レジスタr, アドレスa [,x]

SUBA レジスタr, アドレスa [,x]

- レジスタ間の加算(減算) $r1 = r1 + r2$,
レジスタとメモリの間の加算 $r1 = r1 + *(a+x)$
- 実行するとr1の中身は変更される.
- 例) ADDA R0, A, ; R0=R0+29
- ADDA アドレス, レジスタはないので注意

フラグ



- 演算結果によってセットされる. 条件分岐の条件として利用.
 - SF (Sign Flag): 負の時1, それ以外0
 - ZF (Zero Flag): ゼロの時1, それ以外0
 - OF (Overflow Flag): 演算結果が[-32768, 32767]に収まらない時1, それ以外0.
 - 例) 3-8 SF=1, ZF=0, OF=0
 5-5 SF=0, ZF=1, OF=0
 7fff + 1 SF= , ZF= , OF=
 - ADDAなどの算術命令だけでなく, LDなどでもセットされる.

条件分岐命令

■ 比較命令 ComPare Arithmetic

CPA レジスタr, アドレスa [,x]

□ $r ← *(a+x)$ を実行して, フラグをセットする.

■ 条件分岐

JUMP アドレスa [,x]

JPL アドレスa [,x]

JMI アドレスa [,x]

JZE アドレスa [,x]

□ Jump(無条件), 正(Plus), 負(Minus), 零(Zero)の時にアドレスにジャンプする.

データ格納命令

- 定数 DC (Data Constant)

DC 値

- 値をメモリに格納. #n は16進数.

- メモリ予約 DS (Data Store)

DS *n*

- *n*バイト分メモリを予約. 初期値0で埋める

- 例) DS 3

は, DC 0, DC0, DC0と同じ.

例) 総和を求める

PGM	START	BEGIN	COUNT	DC	3
BEGIN	LAD	GR0,0	A	DC	3
	LAD	GR1,0		DC	6
LOOP	ADDA	GR0,A,GR1		DC	8
	LAD	GR1,1,GR1	B	DS	1
	CPA	GR1,COUNT		END	
	JMI	LOOP			
	ST	GR0,B			
	RET				

例) 総和を求める

0000	PGM	START	BEGIN
0000	BEGIN	LAD	GR0, 0
0002		LAD	GR1, 0
0004	LOOP	ADDA	GR0, A, GR1
0006		ADDA	GR1, ONE
0008		CPA	GR1, COUNT
000A		JMI	LOOP
000C		ST	GR0, B
000E		RET	

```
(Java)
GR0=0;
for(GR1=0;
    GR1 - Count < 0;
    ++GR1){
    GR0 = GR0 + A[GR1];
}
B[0] = GR0;
```

問1

- 次のプログラムを実行した時の、レジスタ GR0 の値とメモリの変化を示せ.

```
PRG   START BEGIN           A      DC      #25
BEGIN LD      GR0,A         B      DC      #22
      CPA     GR0,B         C      DS      1
      JPL     L1
      LD      GR0,B
L1    ST      GR0,C
      RET
```

問2

- N個の最小値を求める次のプログラムを完成させよ.

PRG	START	BEGIN		RET	
BEGIN	LAD	GR1,A	A	DC	#13
	LAD	GR2,0		DC	#32
	LD	GR0,A		DC	#8
LOOP	CPA	GR0,0,GR1		DC	#11
	<u>ア</u>	NEXT	N	DC	4
	LD	GR0,0,GR1	MIN	DS	1
NEXT	LAD	GR1,1,GR1		END	
	LAD	<u>イ</u>			
	CPA	<u>ウ</u>			
	JMI	LOOP			
	ST	GR0,MIN			

問3

- 次のプログラムを実行した時の、レジスタ GR0, GR1, GR2, GR3の値を示せ.

```
PGM      START  BEGIN
BEGIN    LD      GR0, A
          LAD     GR1, A
          LD      GR2, 1, GR1
          LAD     GR3, 1, GR1
          RET
A        DC      #27
B        DC      #1D
C        DS      1
          END
```

宿題

- 4章

- 問1

- 問2

- 問6

- 問7

まとめ

- プロセッサは、主記憶と()間でデータ転送を行う。算術演算は()で実行される。
- CASLは仮想機械COMETの()である。命令コードを()といい、アドレスやレジスタなどの引数を()という。
- メモリの位置を示すには(), ()によるアドレス()などの方法がある。