

第7回 COMET- II のプログラミング

7-1. 高級言語 vs アセンブリ言語

a) 高級言語 (Java とか C とか) と アセンブリ言語との違いを説明してください。

	高級言語 (Java とか)	アセンブリ言語
実行までの手順		
記述レベル (人間に近いとか)		
記述レベルによる メリット/デメリット		

b) COMET- II の書き方について、次の空欄を埋めてください。

- ・ 1行に(.....)
- ・ 1行の構成は ( ..... )
- ・ ラベルは ( ..... )を表し、(.....)のときに使う
- ・ (.....)は、命令の種類を表す
- ・ ( オ..... )は、命令の(.....)を表す

c) 次の命令の並びがあるとき、1行ずつ動作を説明し、全体で何ができたかを説明してください。

LD	GR3,	201	①
ADDA	GR3,	202	②
ST	GR3,	203	③

①の行 .....

.....

②の行 .....

.....

.....

③の行 .....

.....

.....

全体で何ができたか .....

.....

.....

7-2. 計算式と代入 (スライド7-2)

「計算式と代入」の20枚目からの例題  $w=x+y+z$  の作成の手順を追ってみましょう

- ① まず前提として、変数  $w, x, y, z$  はメモリ中の 500, 501, 502, 503 番地の領域においてあるとします。
- ② 計算は、汎用レジスタ GR3 を使うことにしましょう。

x	500 番地
y	501 番地
z	502 番地
w	503 番地

メモリ

...
3
1
2
0
...

a) 手順を(ことばで)書き出してみてください。

1)  $x \Rightarrow GR3$  .....

.....

2) .....

.....

b) 次に、a)で作った手順を、プログラムの命令に置きなおしてみましょう。

1)  $x \Rightarrow GR3 \quad \Rightarrow \Rightarrow \Rightarrow$

2)

c) 「計算式と代入」のスライドの34枚目から、別のプログラムの作り方が載っています。これを見て、上記a)のように手順を書き出したうえで、b)のようにプログラムに置きなおしてください。

1)  $x \Rightarrow GR3 \quad \Rightarrow \Rightarrow \Rightarrow$

2)

d) 上記のb)とc)を比較して言えることを、整理してください。

### 7-3. 条件分岐とIF文 (スライド7-3)

a) 次のJavaのIF文を、フローチャート(流れ図)に書き直してください。 `if (x <= 0) x = -x`

b) CPA命令と条件付ジャンプ命令(JPL、JMI、JZE、JNZ)は何をする命令かを説明してください。

`CPA GR3, GR4` は、GR3とGR4を( )して、もし( )なら、( )する。

`JPL 345` は、( )ならば( )し、そうでなければ( )する。

この2つを続けて実行すると、

GR3とGR4を( )して、もし( )なら、( )し、

そうでなければ( )。

(注意) CPAの後に、別の命令(演算やロードストアなど)を実行すると、フラグの値は別の命令でも書き換わってしまうので、条件ジャンプ命令はCPAの直後に実行するのが望ましい(というか、ほぼ必須)。

### 7-4. 条件分岐2 (スライド7-4)

a) 上記7-3のa)で書いたフローチャートを、COMET-IIのプログラムに直してください。

b) 下記のプログラムを、アセンブリ言語に直してください。

```
if (x>0) {  
    x = x + x  
} else {  
    x = 0  
}
```

7-5. for ループを書く

a) 下記のプログラムの流れ図を描いてください。

```
for (i=0; i<10; i++) {  
    x = x + I;  
}
```

b) 上記の流れ図をアセンブリ言語のプログラムに直してください。

7-6. 配列と指標アドレッシング

a) 指標アドレッシング(インデックスアドレッシング)の復習:

( ..... ) + ( ..... ) ⇒ ( ..... )

b) 次の命令の動作を説明してください。

```
LD GR3, 2345, GR5
```

c) 配列は、同じ型の変数が複数並んでいるもので、配列 A の第 4 番目の要素を A[3] のように書きます。(A[0]から始まるので、A[3]は先頭から 4 番目になる。) 普通は下図のようにメモリ上に連続して要素を置くので、メモリアドレスで見ると連続した場所になっています。

配列A	
100	A[0]
101	A[1]
102	A[2]
103	A[3]
104	A[4]

図のように配列 A の先頭が 100 番地からだとすると、A[i]のアドレスは( ..... ) になります。この動作を、指標アドレッシングを使うとうまく書けます。A[i]に入っている内容を汎用レジスタ GR3 に取り出したいとすると、ロード命令は

LD ..... , 100

と書くことができます。ただしインデックス  $i$  に使うレジスタは GR5 ということにしておきます。こうすると、命令の中のアドレス部分  $100$  を書き換えなくても、GR5 に含まれる  $i$  の値を変えることで、配列中の好きな要素をアクセスすることができます。これが指標アドレッシングの効用です。

d) 次のプログラムの動作を、命令ごとに書き出し、全体として何をするプログラムかわかるように Java 言語で等価なプログラムを書いてください。

但し、配列 A は 100 番地から始まる連続領域だとします。また、400 番地には値 0 が、401 番地には値 1 が、402 番地には値 9 が入っているものとします。

定数 ZERO  $\Rightarrow$  400 番地、定数 ONE  $\Rightarrow$  401 番地、

定数 NINE  $\Rightarrow$  402 番地、ということです。

また、200 番地には変数 S が置いてある(最後に置く)ものとします。

汎用レジスタ GR4 は変数 S のために使い、GR3 はインデックス  $i$  のために使います。GR5 は一時保存のために使っています。

```

LD GR4, 400
LD GR3, 400
L1  CPA GR3, 402
    JPL L2
    LD GR5, 100, GR3
    ADDA GR4, GR5
    ADDA GR3, 401
    JUMP L1
L2  ST GR4 200

```

e) 配列 A[0]~A[9] を配列 B[0]~B[9] にコピーするプログラムを COMET-II/CASL 上で作ってください。配列 A は 100 番地から、配列 B は 200 番地から始まるものとします。定数やレジスタは適当に使ってください。

f) LAD 命令と LD 命令の動作の違い(結果として GR3 に入る値の違い)を説明してください

LD GR3, 200 の結果 .....

LAD GR3,, 200 の結果 .....