

コンピュータ ハードウェア (第11回:2012年1月17日)

アセンブリ言語 乗算とサブルーチン

小嶋 秀樹

xkozima@myu.ac.jp

この授業でめざすもの

(1) 論理素子と
論理回路について学ぶ

回路シミュレータ
「らくらくロジック」

(2) コンピュータの
基本構造を理解する

2進数の扱いを
マスター

COMET II
CASL II

(3) マイクロプロセッサと
アセンブリ言語を学ぶ

シミュレータ
「WCASL II」

(4) C言語における
メモリの扱いを学ぶ

授業スケジュール(予定)

- 10月 4日 イン트로ダクション
- 10月11日 論理素子を動かす
- 10月18日 論理回路をつくる
- 10月25日 休講: 海外出張(学会)のため
- 11月 1日 真偽値表から論理回路へ
- 11月 8日 休講: 海外出張(講演会)のため
- 11月15日 論理回路の最適化
- 11月22日 論理回路と二進数
- 11月29日 休講: 海外出張(学会)のため
- 12月 6日 論理回路と記憶・計算
- 12月13日 マイクロプロセッサとメモリ
- 12月20日 アセンブリ言語: 加算と減算
- 1月10日 アセンブリ言語: 条件分岐と繰り返し
- 1月17日 アセンブリ言語: 乗算とサブルーチン
- 2月 7日 試験日: 最終小テストを行なう予定

1月24日は木曜扱い
1月31日は金曜扱い

前回の復習

アセンブリ言語

条件分岐と繰り返し

アセンブリ言語と機械語(マシン語)

The screenshot shows the WCASL-II [COMET Simulator] interface. On the left is a CPU diagram with components: IR (FFFF FFFF), PR (0020), SP (00AB), Decoder (0000 0000), Controller, GR (0-7, all FFFF), Adder, MAR (FFFF), MDR (FFFF), FR (000), and ALU. On the right is a table with two columns: **機械語** (Machine Code) and **アセンブリ言語** (Assembly Language). The assembly code is divided into **プログラム** (Program) and **データ** (Data) sections. A red arrow points to the assembly code with the text: アセンブリ言語を機械語に変換するのが「アセンブラ」.

機械語	アセンブリ言語
0020 1010	LD GR1,A
0021 0027	
0022 2010	ADDA GR1,B
0023 0028	
0024 1110	ST GR1,C
0025 0029	
0026 8100	RET
0027 005D	DC 93
0028 0061	DC 97
0029 0000	DC 0
002A FFFF	
002B FFFF	
002C FFFF	
002D FFFF	
002E FFFF	
002F FFFF	
0030 FFFF	
0031 FFFF	
0032 FFFF	
0033 FFFF	
0034 FFFF	
0035 FFFF	
0036 FFFF	

(1) LD GR1,LABEL

LABELで参照されるメモリの内容をGR1に代入(メモリ内容是不変)

$GR1 \leftarrow (LABEL)$

(2) ST GR1,LABEL

GR1の内容をLABELで参照されるメモリに代入(GR1是不変)

$LABEL \leftarrow (GR1)$

(3) ADDA GR1,LABEL

GR1の内容とLABELの内容を加算してGR1に代入(メモリ是不変)

$GR1 \leftarrow (GR1) + (LABEL)$

(4) ADDA GR1,GR2

GR1の内容とGR2の内容を加算してGR1に代入(GR2是不変)

$GR1 \leftarrow (GR1) + (GR2)$

小テストをしました

つぎのプログラムは, DATAの値(44)に何らかの演算を行ない, その結果をRESに格納します. 得られるRESの値を10進数で答えなさい. また演算の意味を説明しなさい.

```
TST    START
        LD     GR1,DATA
        XOR    GR1,TMP1
        ADDA   GR1,TMP2
        ST     GR1,RES
        RET

DATA    DC     44
TMP1    DC     #FFFF
TMP2    DC     1
RES     DC     0
        END
```

GR1 に 44 を代入
#FFFF との XOR を計算
その結果に1を加算
その結果を RES に格納

「全ビットを反転させてから
1を加える」

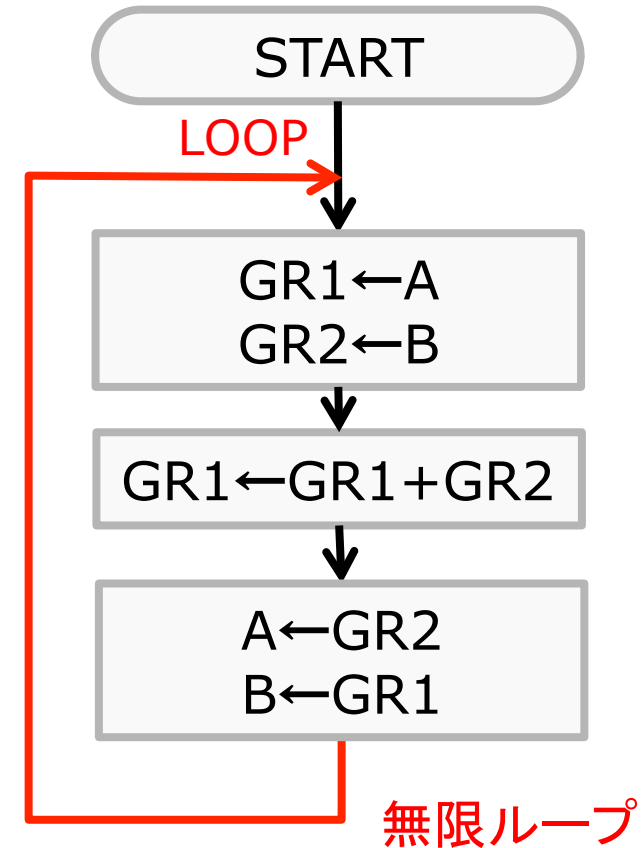
「44 の『2の補数』を計算」

「-44 を『2の補数』で得る」

無条件ジャンプ

JUMP <ラベル>

```
FIB      START
LOOP    LD      GR1,A
        LD      GR2,B
        ADDA   GR1,GR2
        ST      GR2,A
        ST      GR1,B
        JUMP   LOOP
A       DC      1
B       DC      1
        END
```

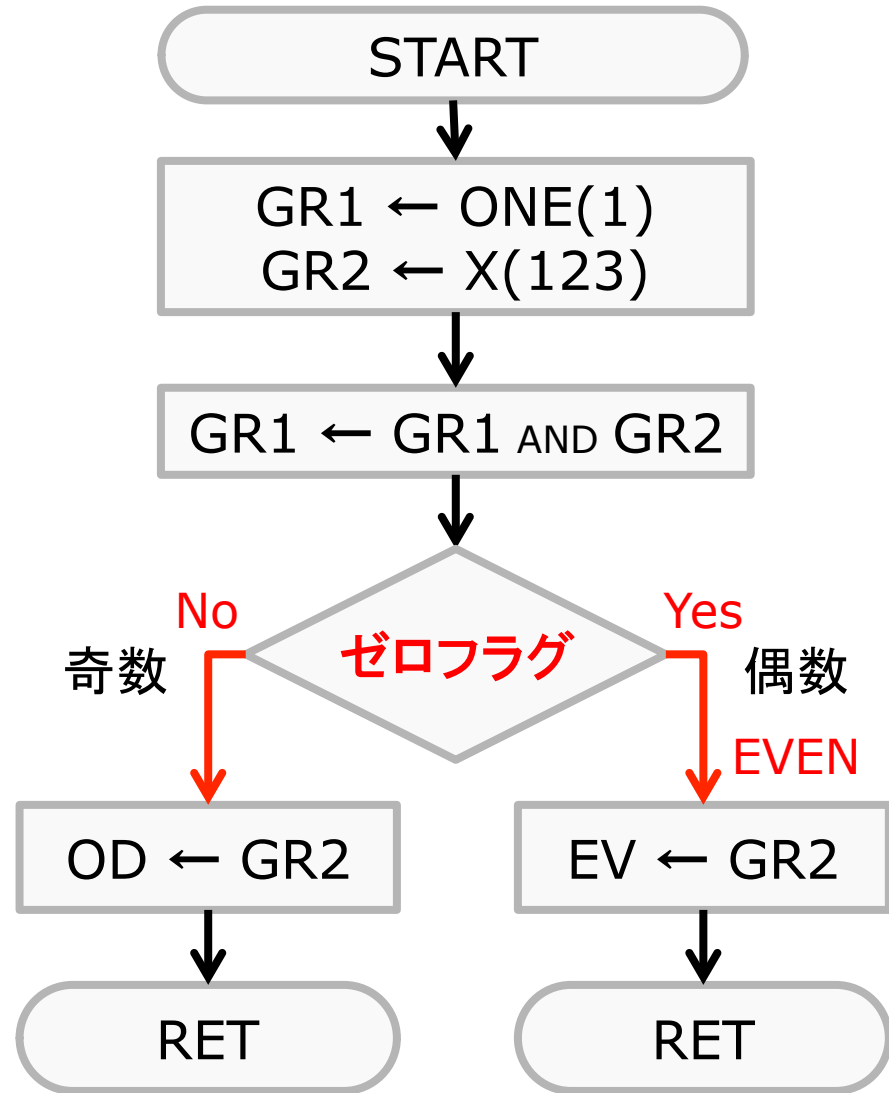


A: 1, 1, 2, 3, 5, 8, 13, 21, 34, ... }
B: 1, 2, 3, 5, 8, 13, 21, 34, 55, ... } → 黄金比 1:1.618
(フィボナッチ数列)

条件ジャンプ(条件分岐)

PAR	START	
	LD	GR1,ONE
	LD	GR2,X
	AND	GR1,GR2
	JZE	EVEN
	ST	GR2,OD
	RET	
EVEN	ST	GR2,EV
	RET	
ONE	DC	1
X	DC	123
EV	DC	0
OD	DC	0
	END	

直前の「結果」でジャンプ/素通り

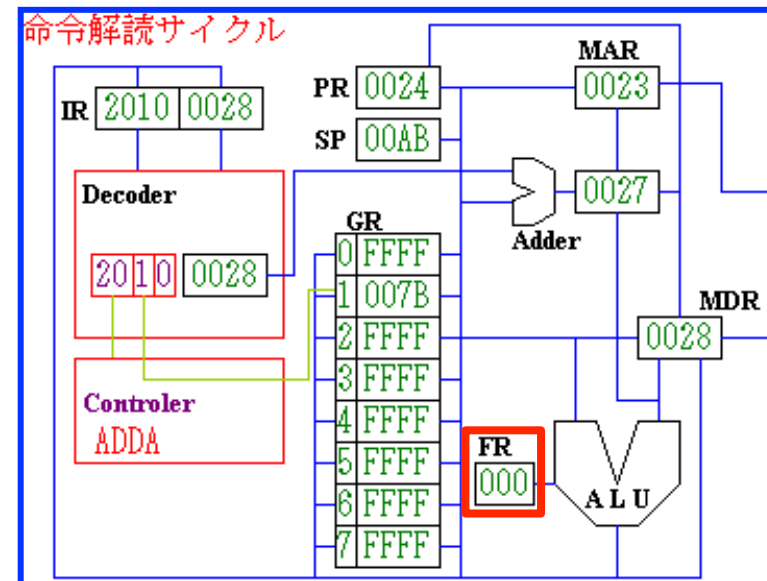
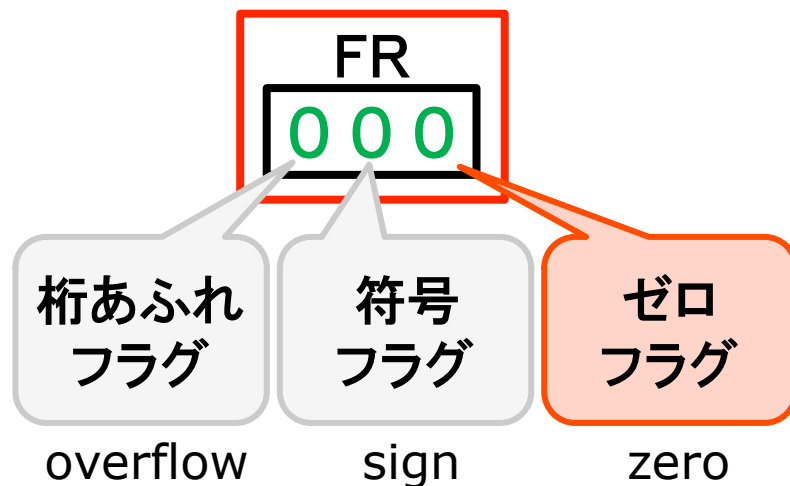


条件ジャンプ(条件分岐)

直前の「結果」でジャンプ/通過

JZE 〈ラベル〉	ゼロ結果(ゼロフラグ=1)のときジャンプ
JNZ 〈ラベル〉	非ゼロ結果(ゼロフラグ=0)のときジャンプ
JPL 〈ラベル〉	正結果(符号フラグ=0)のときジャンプ
JMI 〈ラベル〉	負結果(符号フラグ=1)のときジャンプ
JOV 〈ラベル〉	桁あふれ(桁あふれフラグ=1)のときジャンプ

フラグ レジスタ (FR)



繰り返し

小テストをやりました

$N + \dots + 3 + 2 + 1$ を計算する
プログラムを CASL II で書いてください。
($N=5$ について動作を確かめてください)

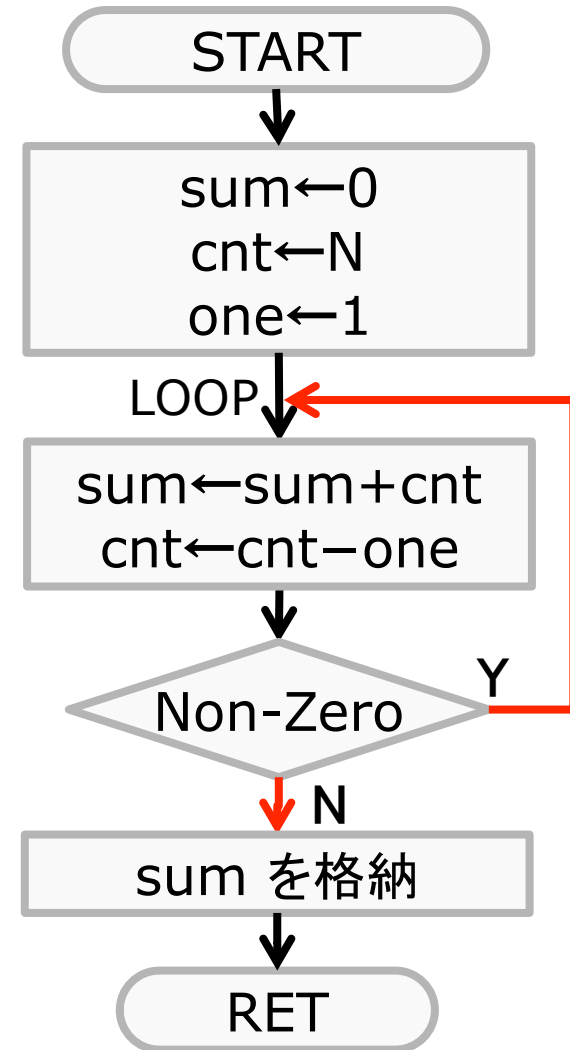
ヒント
(1)

```
sum = 0;
for (cnt=N; cnt>0; cnt=cnt-1) {
    sum = sum+ cnt;
}
```

ヒント
(2)

```
sum ← 0
cnt ← 5
one ← 1
LOOP sum ← sum + cnt
      cnt ← cnt - one
      JNZ LOOP
      ...
```

ヒント
(3)



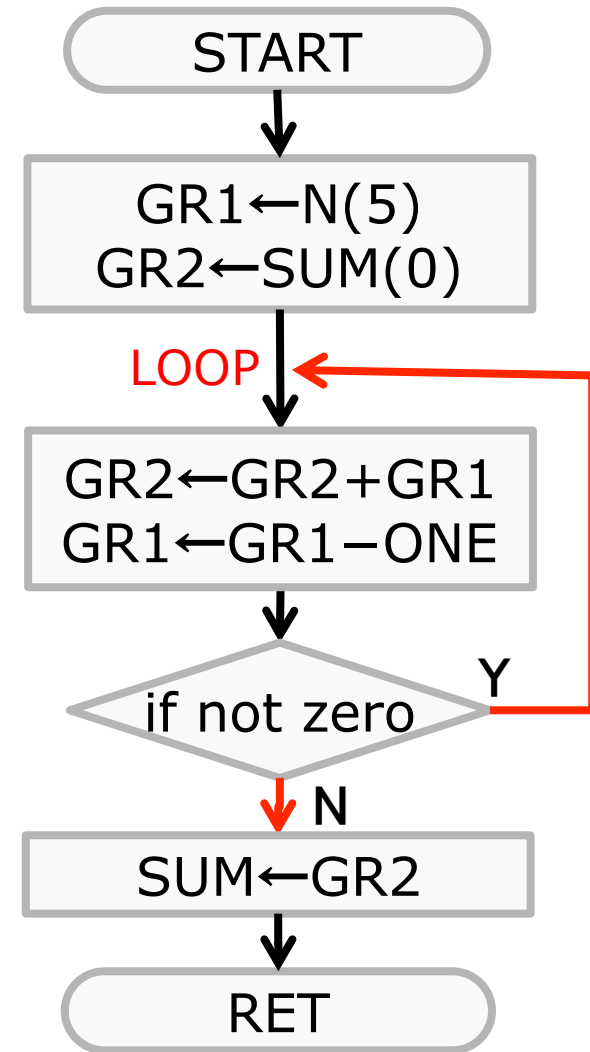
繰り返し

小テストをやりました(解答例)

$N + \dots + 3 + 2 + 1$ を計算する
プログラムを CASL II で書いてください。
($N=5$ について動作を確かめてください)

```
TEST START
      LD   GR1,N
      LD   GR2,SUM
LOOP  ADDA GR2,GR1
      SUBA GR1,ONE
      JNZ LOOP
      ST   GR2,SUM
      RET

N     DC   5
ONE   DC   1
SUM   DC   0
      END
```



(1) JUMP LABEL

LABELで参照される場所にジャンプ

PR ← LABEL (プログラムレジスタの書き換え)

(2) JZE LABEL

直前の演算結果が**ゼロ**のとき

LABELで参照されるメモリに代入 (GR1は不変)

PR ← LABEL

そうでなければ素通り

(3) JNZ LABEL

直前の演算結果が**ゼロでない**とき

LABELの内容を加算してGR1に代入 (メモリは不変)

PR ← LABEL

そうでなければ素通り

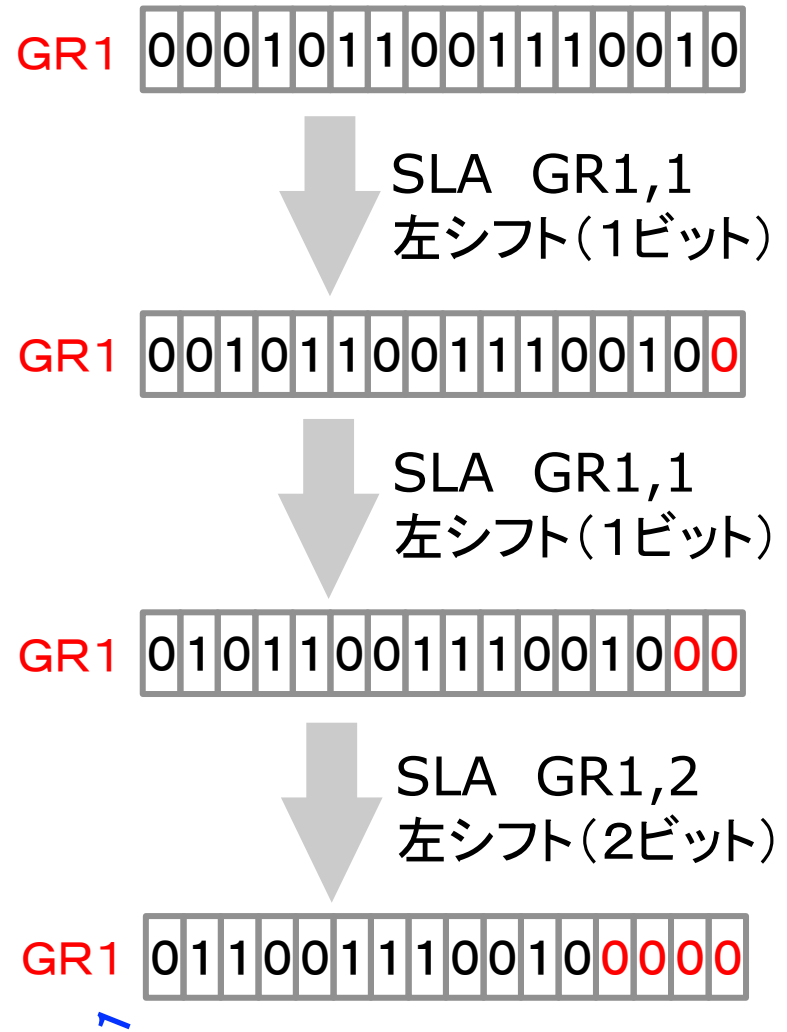
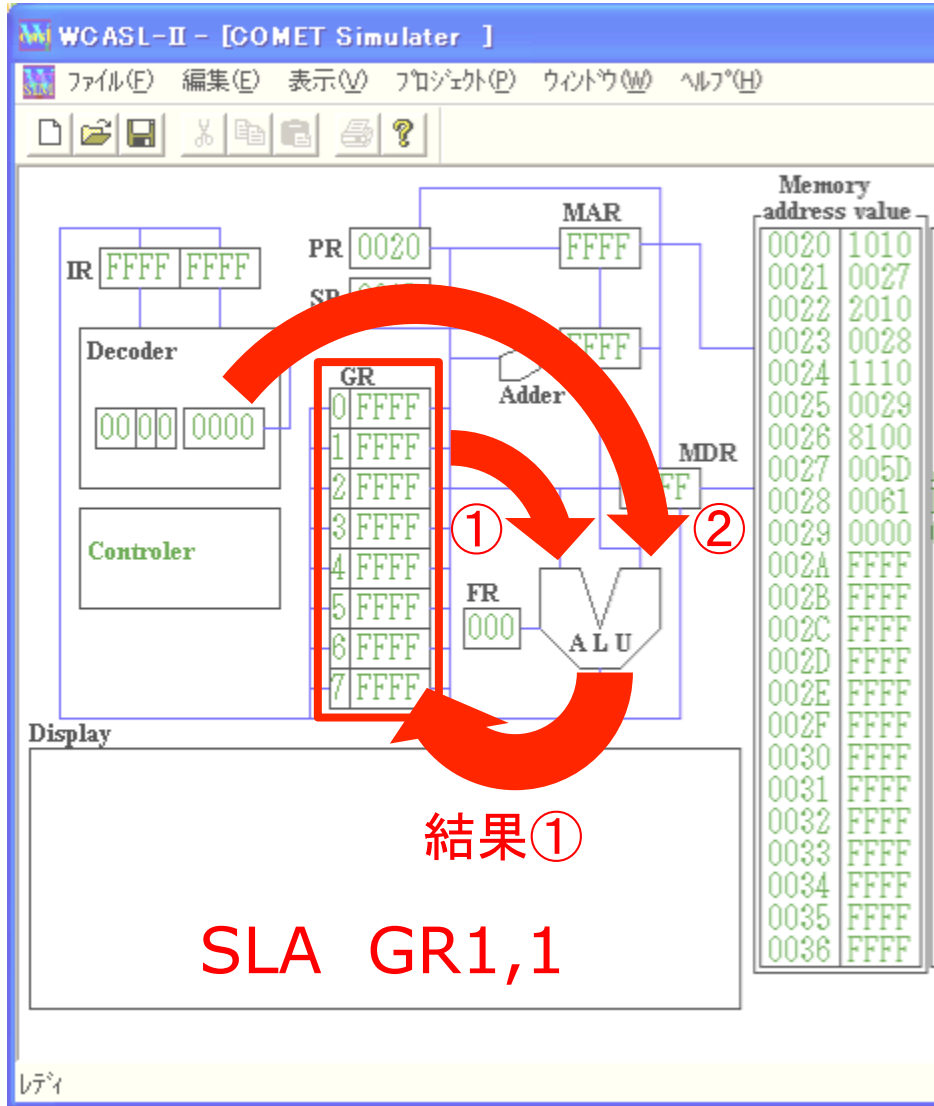
今回のテーマ(1)

アセンブリ言語

シフト演算

シフト演算命令

SLA: 左シフト(2倍)



シフト演算命令

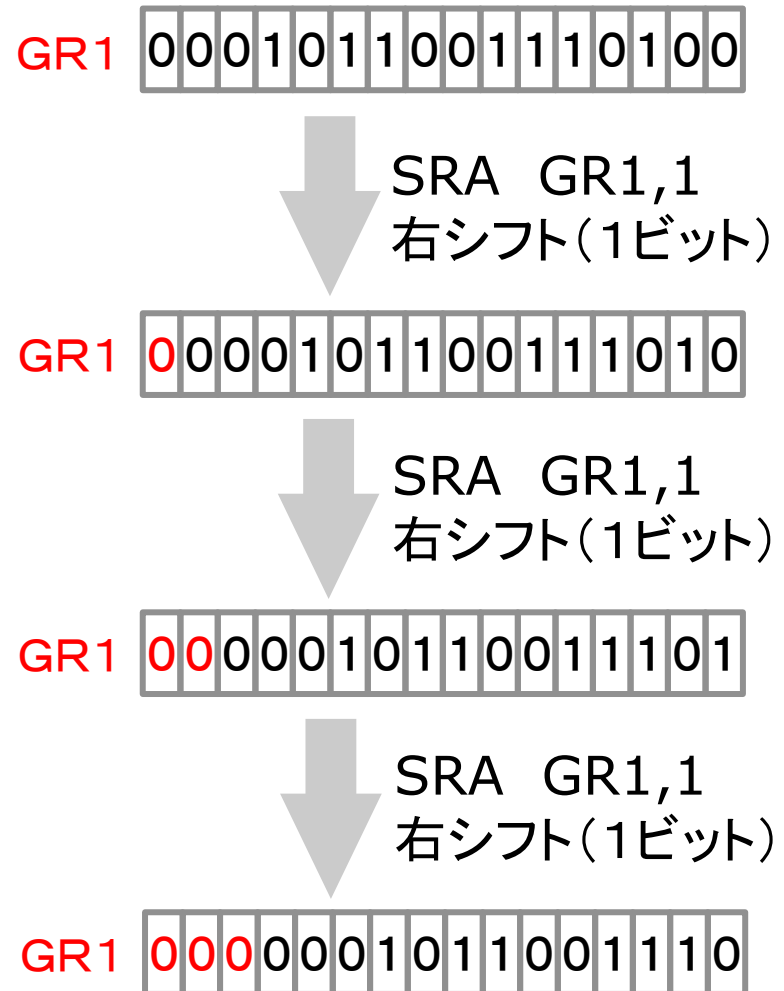
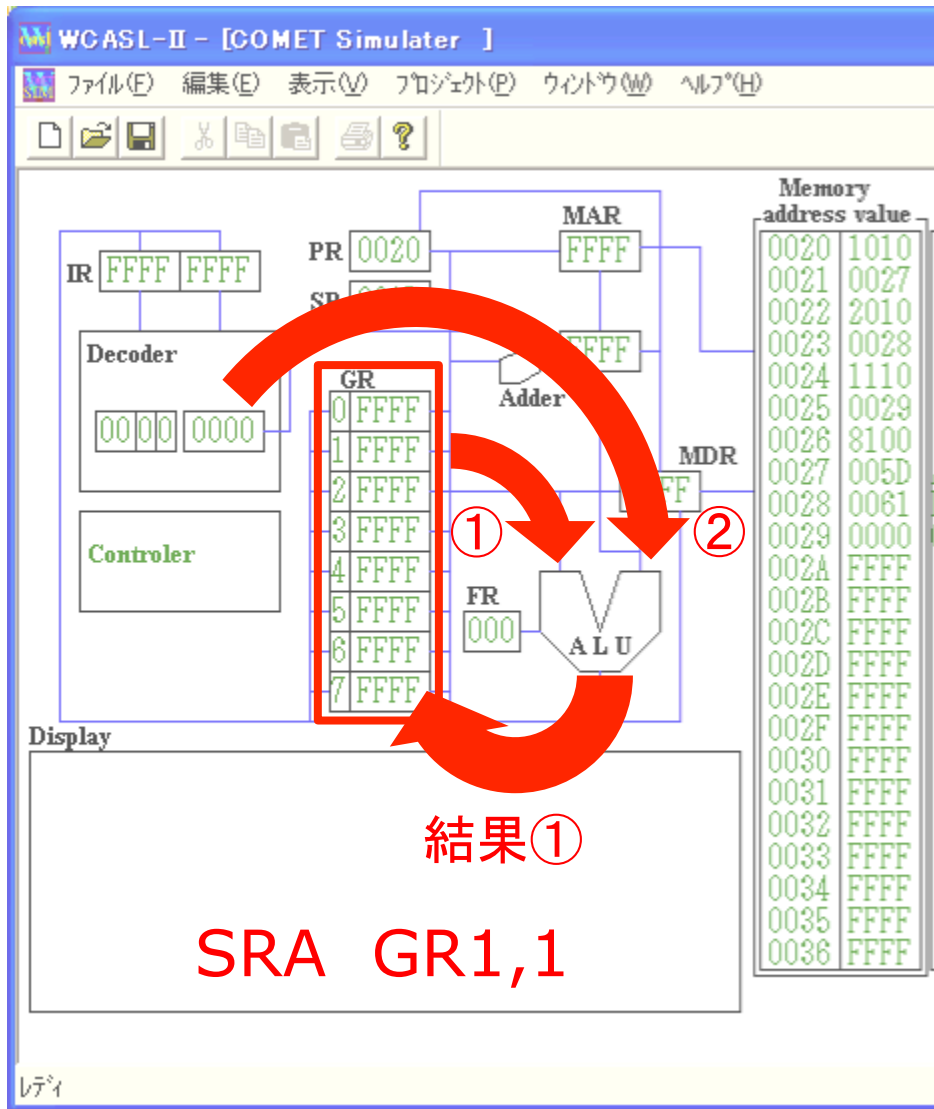
SLA (Shift Left Arithmetic)

```
KOZ  START
      LD   GR1,A    ;GR1←A
      SLA  GR1,2    ;GR1を左シフト(2ビット)
      ST   GR1,X    ;X←GR1
      RET
A     DC   #04D2    ;1234
X     DC   0        ;結果#1348(=4936)
      END
```

$$\begin{array}{r} \ll 2 \\ \hline \#04D2 = 0000\ 0100\ 1101\ 0010 = 1234 \\ \#1348 = 0001\ 0011\ 0100\ 1000 = 4936 \\ \hspace{15em} (4倍) \end{array}$$

シフト演算命令

SRA: 右シフト(1/2倍)



シフト演算命令

SRA (Shift Right Arithmetic)

```

KOZ  START
      LD    GR1,A    ;GR1←(A)
      SRA  GR1,1    ;GR1を右シフト(1ビット)
      ST    GR1,X    ;X←(GR1)
      RET
A     DC    #04D2    ;1234
X     DC    0        ;結果#0269(=617)
      END
  
```

$$\begin{array}{r}
 >>1 \quad \#04D2 = & 0000 & 0100 & 1101 & 0010 & = & 1234 \\
 \hline
 & \#0269 = & 0000 & 0010 & 0110 & 1001 & = & 617 \\
 & & & & & & & (1/2倍)
 \end{array}$$

今回のテーマ(2)

アセンブリ言語

乗算

乗算の復習

$123_{(10)} \times 356_{(10)}$ の計算方法

$$123 \times 6 \times 1 = 738$$

$$123 \times 5 \times 10 = 6150$$

$$123 \times 3 \times 100 = 36900$$

43788

$0111_{(2)} \times 0101_{(2)}$ の計算

$$0111 \times 1 \times 00001 = 00000111$$

$$0111 \times 0 \times 00010 = 00000000$$

$$0111 \times 1 \times 00100 = 00011100$$

$$0111 \times 0 \times 01000 = 00000000$$

$00100011 = 35_{(10)}$

これと同じ手順を
2進数について
実行すればよい

シフトと加算を
組み合わせる

乗算の復習

A=7 B=5
0111₍₂₎ × 0101₍₂₎

GR2(A)	GR1(B)
00000111	0101
00000000	0010
00011100	0001
00000000	0000

00100011
GR3(ANS)

AND

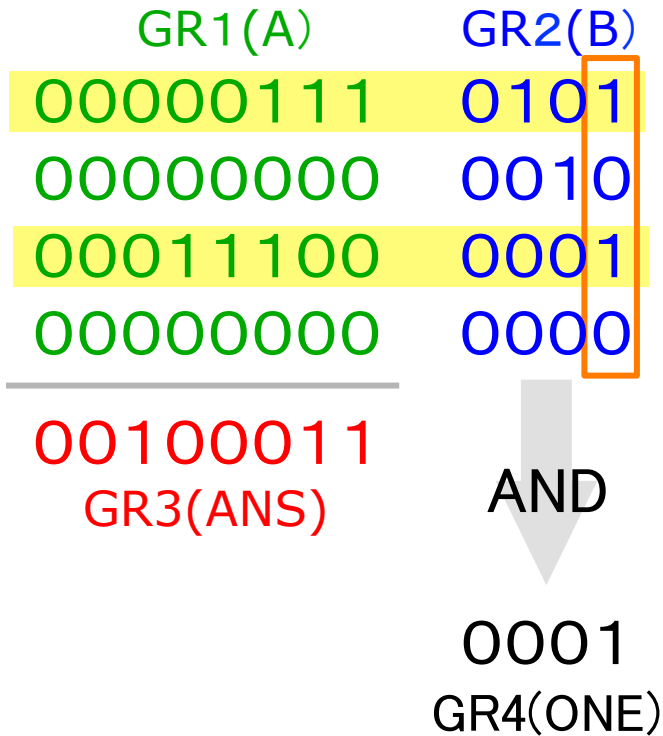
0001
GR4(ONE)

「CASL シミュレーション」
で動作を確認

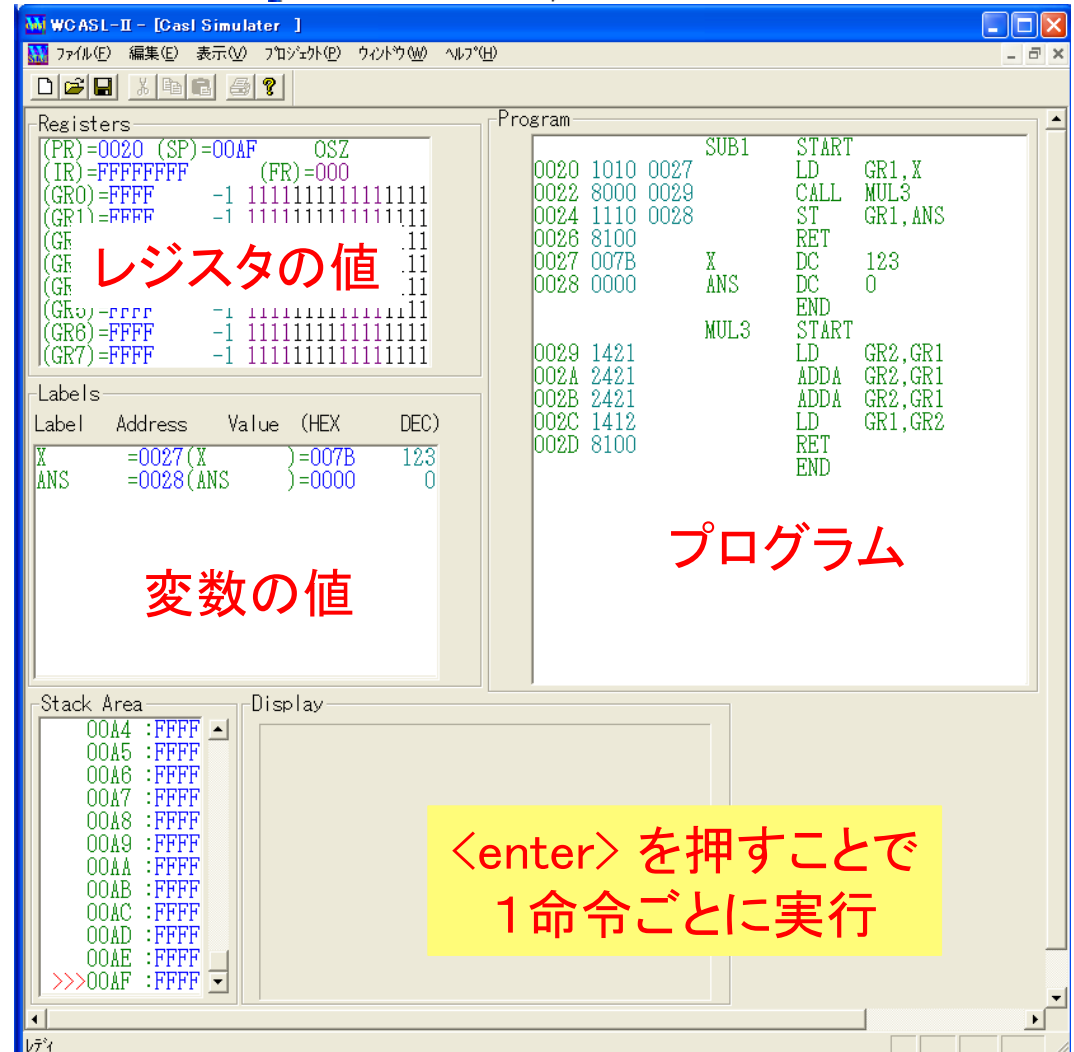
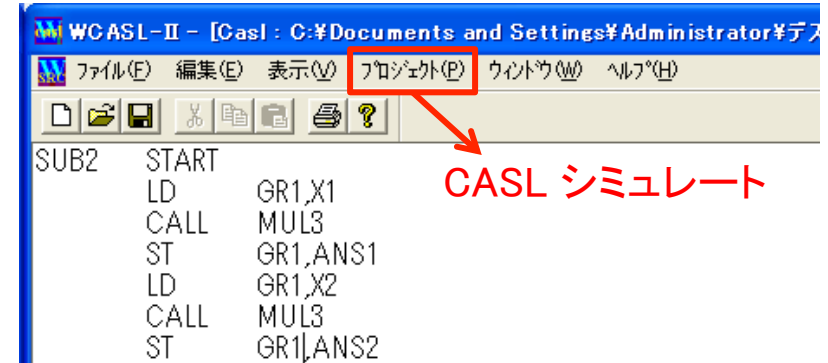
```
MULT      START
          LD      GR1,B
          LD      GR2,A
          LD      GR3,ANS
LOOP      LD      GR4,ONE
          AND     GR4,GR1
          JZE     SKIP
          ADDA    GR3,GR2
SKIP      SLA     GR2,1
          SRA     GR1,1
          JNZ     LOOP
          ST      GR3,ANS
          RET
ANS      DC      0
A        DC      7
B        DC      5
ONE      DC      1
          END
```

乗算の復習

A=7 B=5
 $0111_{(2)} \times 0101_{(2)}$



「CASL シミュレート」
 で動作を確認



今回のテーマ(3)

アセンブリ言語

サブルーチン

サブルーチン

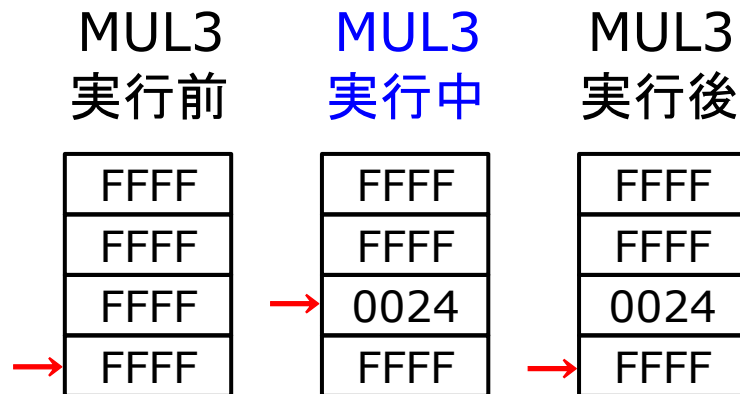
(副プログラム)

CALL MUL3

→ MUL3 に分岐.

RET で戻る.

戻るべき位置(アドレス)を
スタックに記録しておく.



```
SUB1    START
         LD    GR1,X
         CALL  MUL3
         ST    GR1,ANS
         RET
X       DC    123
ANS     DC    0
         END

;
MUL3    START
         LD    GR2,GR1
         ADDA  GR2,GR1
         ADDA  GR2,GR1
         LD    GR1,GR2
         RET
         END
```

サブルーチン

(副プログラム)

CALL MUL3

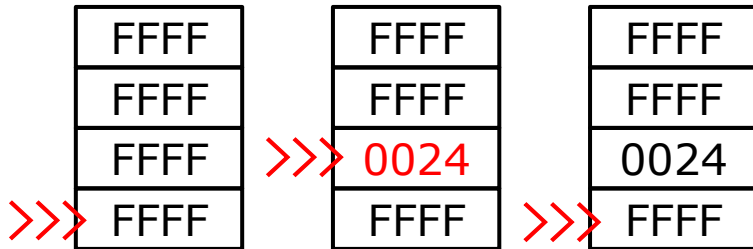
→ MUL3 に分岐.
RET で戻る.

戻るべき位置(アドレス)を
スタックに記録しておく.

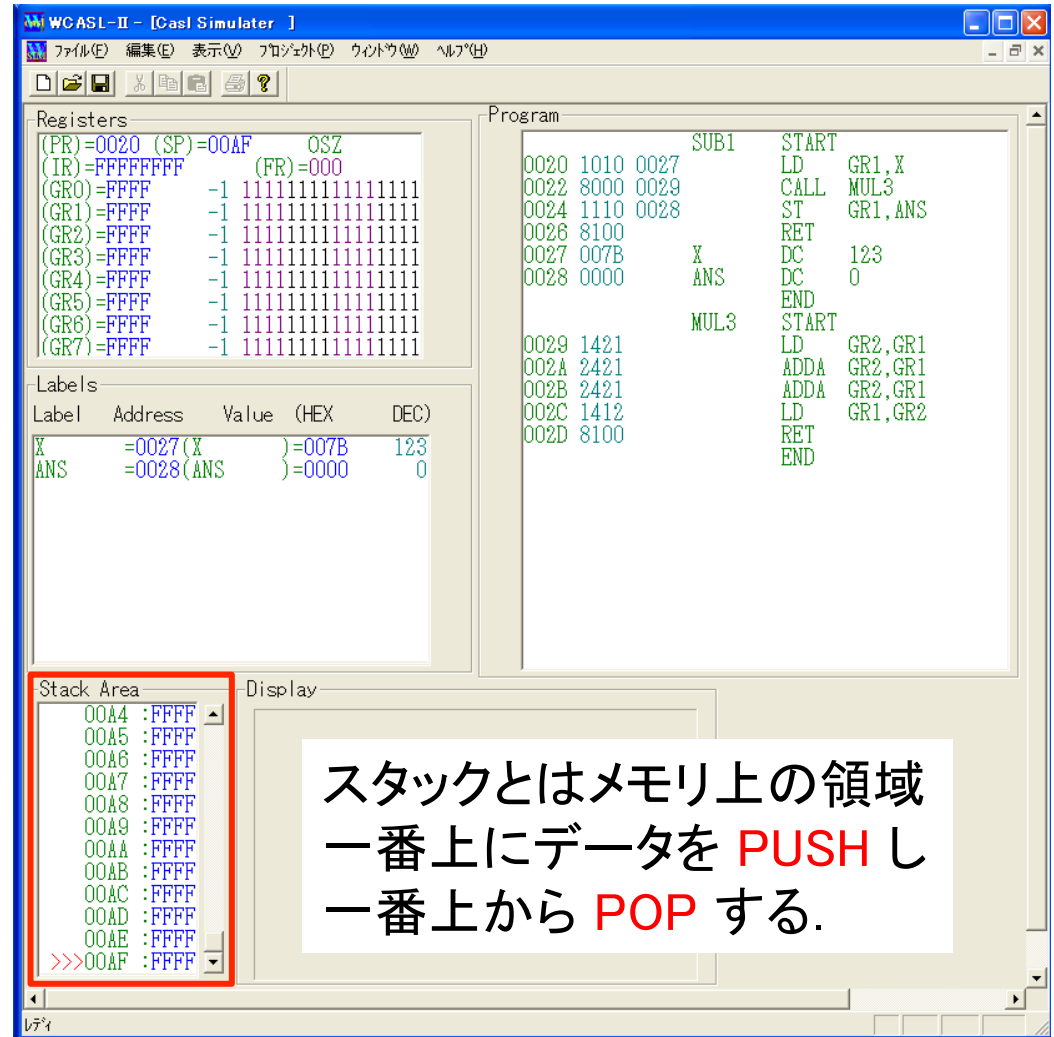
MUL3
実行前

MUL3
実行中

MUL3
実行後



→ : SP(スタックポインタ)



スタックとはメモリ上の領域
一番上にデータを **PUSH** し
一番上から **POP** する.

スタックは
下(アドレス大)から
上(アドレス小)へ伸びる

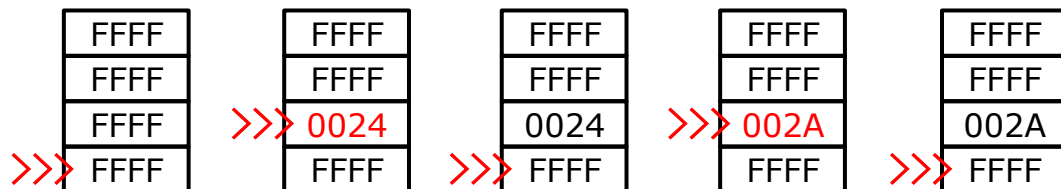
サブルーチン

(副プログラム)

ある機能をもつサブルーチンを再利用 (CALL MUL3)

データのやりとりには汎用レジスタを使う (GR1)

ただし、サブルーチン内部でいくつかの汎用レジスタを上書きしてしまう (GR2)



```
SUB2      START
          LD      GR1,X1
          CALL    MUL3
          ST      GR1,ANS1
          LD      GR1,X2
          CALL    MUL3
          ST      GR1,ANS2
          RET
          DC      123
          DC      999
          DC      0
          DC      0
          END
;
MUL3     START
          LD      GR2,GR1
          ADDA   GR2,GR1
          ADDA   GR2,GR1
          LD      GR1,GR2
          RET
          END
```

サブルーチン

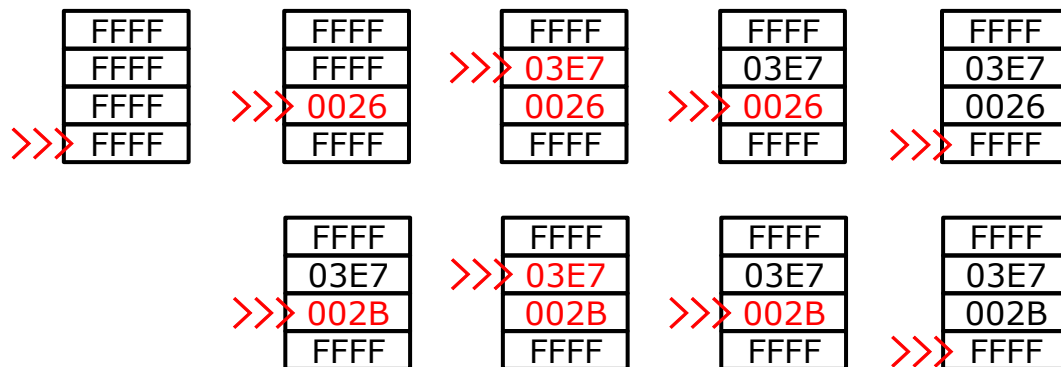
(副プログラム)

PUSH 0,GR2

GR2 の値を
スタックに積む(プッシュ)

POP GR2

スタックのトップから
値を取り出し(ポップ)
それを GR2 に格納



```
SUB2          START
              LD      GR1,X1
              LD      GR2,X2
              CALL    MUL3
0026          ST      GR1,ANS1
              LD      GR1,GR2
              CALL    MUL3
002B          ST      GR1,ANS2
              RET
X1            DC      123
X2            DC      999
ANS1          DC      0
ANS2          DC      0
              END
```

```
;
MUL3         START
              PUSH    0,GR2
              LD      GR2,GR1
              ADDA   GR2,GR1
              ADDA   GR2,GR1
              LD      GR1,GR2
              POP     GR2
              RET
              END
```

次回は 2月7日 (最終回)
(後半に中テストを実施します)

以上です

1月24日は木曜扱い
1月31日は金曜扱い

<http://www.myu.ac.jp/~xkozima/course/hardware.html>

授業で使った資料(スライドなど)は,
ここからダウンロードできるようにします.