プロセッサと機械語

2年次前期　（第5回）

中島 克人

情報メディア学科
nakajima@im.dendai.ac.jp

アセンブリ言語CASLⅡ

- プログラム例（SMPL2）

```
SMPL2   START
        LAD   GR0,#0001        ; #0001 → GR0
        LAD   GR1,1            ; 1 → GR1
        LD    GR2,DAT          ; DAT 番地の内容を GR2 に
                               ; 即ち, <DAT> → GR2
        LD    GR3,DAT,GR1      ; <e>(=<DAT+<GR1>>) → GR3
        ADDA  GR2,GR3          ; <GR2>+<GR3> → GR2
        ST    GR2,ADR1         ; <GR2> → ADR1
        RET
DAT     DC    #0002            ; 定数#0002 をここに配置
        DC    #0003            ; 定数#0003 をここに配置
ADR1    DS    1                ; 1語分の領域 をここに確保
        END
```

CASLシミュレータで実行する前に，結果を予想して見よう！

COMET Ⅱマシンの機械命令

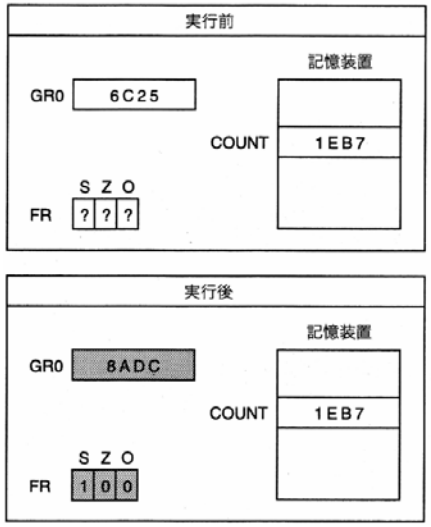
（教科書 p.18）

- 論理加算（add logical）命令

[書き方]　ADDL　gr,adr[,xr]

[機能]　<GRn> ＋ <e> → T;
<T> → GRn;
if <T> ＞ 65535＝$2^{16}-1$
then 1→OF　else 0→OF;
SZ設定
if <GRn>=0 then 1→ZF
else 0→ZF
<GRn[15]>→SF;

[実行例]　ADDL　GR0,COUNT



（1語命令）

[書き方]　ADDL　gr1,gr2

[機能]　上記と同様

COMET Ⅱマシンの機械命令

（教科書 p.18）

- 算術加算ADDAと論理加算ADDLの違い

演算結果（16ビットのビットパターン）は同じ
（ただし，最上位ビットが1の場合，前者は負の数字，後者は正の数字を表している）

オーバフローの解釈が異なる

（例）

```
          0111 1111 1111 1111
ADDA)     0000 0000 0000 0001
          1000 0000 0000 0000
```

別の解釈：

符号ビットへの桁上げ：有
符号ビットからの桁上げ：無
} 故にoverflow

```
          0111 1111 1111 1111
ADDL)     0000 0000 0000 0001
          1000 0000 0000 0000
```

別の解釈：

最上位ビットからの桁上げ：無
} 故にoverflow 無し

アセンブリ言語CASLⅡ

- プログラム例（SMPL3）

GR0 に#01C5 をロードした後，COUNT番地の #1F9A と算術加算し，結果の値とFRの値を確認せよ

```
SMPL3   START
        LAD     GR0,#01C5       ; #01C5 を GR0 に
                                ; <GR0>＋<COUNT> → GR0
COUNT   DC      #1F9A           ; 定数#1F9A をここに配置
        END
```

同様に，GR0 に#6C25 をロードした後，COUNT番地の #1EB7 と算術加算し，結果の値とFRの値を確認せよ

同様に，GR0 に#6C25 をロードした後，COUNT番地の #1EB7 と論理加算し，結果の値とFRの値を確認せよ

COMET Ⅱマシンの機械命令

- 算術減算（subtract arithmetic）命令 （教科書 p.19）

[書き方] SUBA gr,adr[,xr]

[機能] <GRn> − <e> → T;
<T> → GRn;
if (<T> ＞ 32767 ＝$2^{15}-1$
or <T> ＜ −32768)
then 1→OF else 0→OF;
SZ設定
if <GRn>=0 then 1→ZF
else 0→ZF
<GRn[15]>→SF;

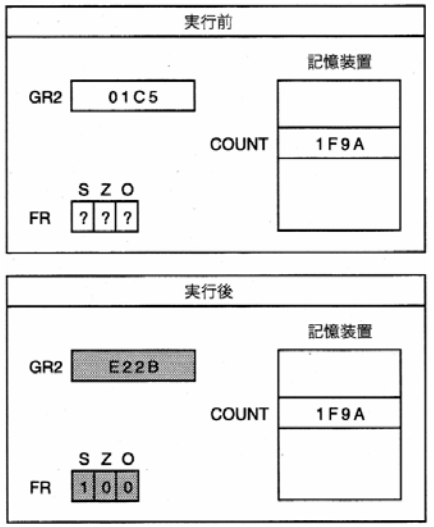
もし，16ビットでの2の補数表現の範囲を超えていると

（1語命令）

[書き方] SUBA gr1,gr2

[機能] 上記と同様

[実行例] SUBA GR2,COUNT



COMET Ⅱマシンの機械命令

- 論理減算（subtract logical）命令 （教科書 p.19）

[書き方] SUBL gr,adr[,xr]

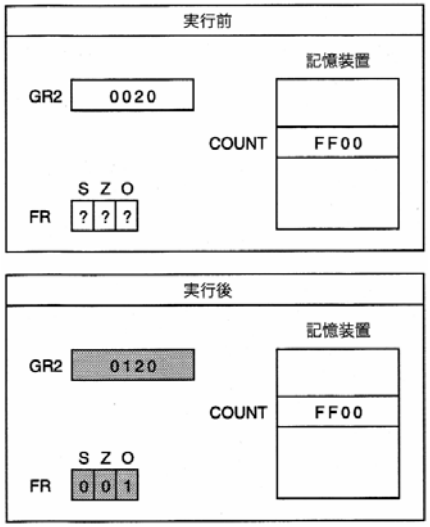
[機能] <GRn> − <e> → T;
<T> → GRn;
if <T> ＜ 0
then 1→OF else 0→OF;
SZ設定
if <GRn>=0 then 1→ZF
else 0→ZF
<GRn[15]>→SF;

（1語命令）

[書き方] SUBL gr1,gr2

[機能] 上記と同様

[実行例] SUBL GR2,COUNT



演習

- 下記を行う SMPL4 を作成し，実行して見よ

DAT番地のデータ #0001 と（DAT＋1）番地のデータ #8000 を算術加算し，（DAT＋2）番地に格納せよ

```
SMPL4   START
        LAD     GR1,1           ; 1 → GR1
                                ; <DAT> → GR2
                                ; <e>(=<DAT＋<GR1>>) → GR3
                                ; <GR2>＋<GR3> → GR2
                                ; 2 → GR1
                                ; <GR2> → DAT＋<GR1>

DAT                             ; 定数#0001 をここに配置
                                ; 定数#8000 をここに配置
                                ; 1語分の領域 をここに確保
        END
```

演習

- SMPL4 の実行の結果，SF, ZF, OF はどうなったか？
  その理由も考察せよ　　FR（フラグレジスタ）

ヒント：

- SZ設定する命令は FR を3ビットまとめて上書きする
- 最後の SZ設定 が有効となる

解答：

- 最後に SZ設定 したのは　　命令で，結果は #
- その時の結果は，SF＝　，ZF＝　，OF＝

- 算術加算の代りに算術減算とすると，SF, ZF, OF はどうなるか？

解答：

-

アセンブリ言語CASL Ⅱ

- 演習SUBL … SUBL を使って，下記を計算するプログラムを作成し，WCASL Ⅱで実行せよ
- 論理減算にて，#80FF － 100 － #35F5 を実行し，答えを16進数で回答せよ
  そして，オーバフローしたかどうかを確かめよ
- 上記と同じ計算を算術減算SUBAにて行い，オーバフローも確かめよ

```
SUBL    START
        LAD     GR1,
        LAD     GR2,100
        SUBL    GR1,GR2
        LAD     GR2,
        ST      GR1,RSLT
        RET
RSLT    DS      1
        END
```

```
SUBA    START
        LAD     GR1,
        LAD     GR2,100
        SUBA    GR1,GR2
        LAD     GR2,
        ST      GR1,RSLT
        RET
RSLT    DS      1
        END
```


アセンブリ言語CASL Ⅱ

- ここまでに学んだ命令

機械語命令（Comet IIマシンが直接解釈し実行する命令）：

- LD (Load)
- LAD (Load Address)
- ST (Store)
- ADDA (Add Arithmetic)
- ADDL (Add Logical)
- SUBA (Subtract Arithmetic)
- SUBL (Subtract Logical)
- RET (Return)

アセンブラ命令（アセンブラに指示するための命令（マシンは実行しない））：

- START …アセンブルすべきプログラムの開始位置を示す
- DC (Define Constant) … 定数を主記憶上に配置する
- DS (Define Storage) … 主記憶上に空き領域を確保する
- END …アセンブルすべきプログラムの最後を示す

プロセッサと機械語

論理演算と比較

COMET Ⅱマシンの機械命令

- 論理演算（0と1からなる論理値同士の演算）

  論理積（AND） ... X and Y
  論理和（OR） ... X or Y
  排他的論理和（xOR） ... X xor Y
  否定（NOT） ... not X

X Y	X and Y	X or Y	X xor Y	not X
0 0	0	0	0	1
0 1	0	1	1	1
1 0	0	1	1	0
1 1	1	1	0	0

- 1bit 演算例

X = 0_2
and) Y = 1_2
X and Y = 0_2

- 2bit 演算例

X = $0\ 1_2$
and) Y = $1\ 1_2$
X and Y = $0\ 1_2$

bit反転
即ち，Y=1の時，結果はXの反対で，
Y=0の時，結果はXと同じ

各bit位置（桁）ごとに
表にある演算規則を
適用する

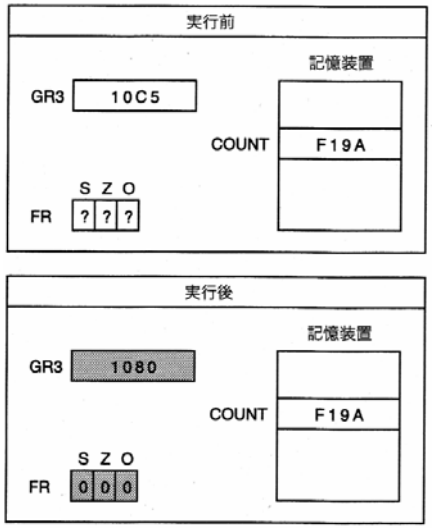
COMET Ⅱマシンの機械命令

（教科書 p.19）

- 論理積（and）命令

[書き方] AND gr,adr[,xr]
[機能] <GRn> ∧ <e> → GRn;
0→OF;
SZ設定
{ if <GRn>=0 then 1→ZF
else 0→ZF
<GRn[15]>→SF;

[実行例] AND GR3,COUNT



（1語命令）
[書き方] AND gr1,gr2
[機能] 上記と同様

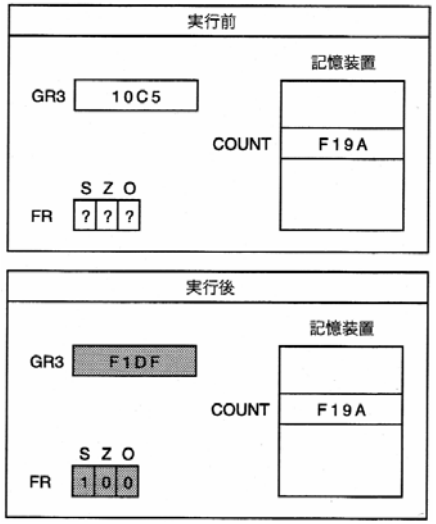
COMET Ⅱマシンの機械命令

（教科書 p.20）

- 論理和（or）命令

[書き方] OR gr,adr[,xr]
[機能] <GRn> ∨ <e> → GRn;
0→OF;
SZ設定
{ if <GRn>=0 then 1→ZF
else 0→ZF
<GRn[15]>→SF;

[実行例] OR GR3,COUNT



（1語命令）
[書き方] OR gr1,gr2
[機能] 上記と同様

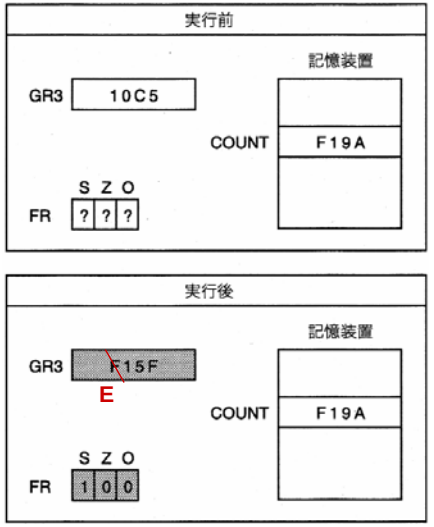
COMET Ⅱマシンの機械命令

（教科書 p.20）

- 排他的論理和（exclusive or）命令

[書き方] XOR gr,adr[,xr]
[機能] <GRn> ⊕ <e> → GRn;
0→OF;
SZ設定
{ if <GRn>=0 then 1→ZF
else 0→ZF
<GRn[15]>→SF;

[実行例] XOR GR3,COUNT



（1語命令）
[書き方] XOR gr1,gr2
[機能] 上記と同様

論理演算の使用例

- AND命令

```
  10100110        11100100
and)00001111    and)11110000
  --------        --------
  00000110        11100000
```

下位4 bit だけを残し，他のbit を 0 にする マスク(mask)

- OR命令

```
  10100110        00000110
or) 00001111    or) 11100000
  --------        --------
  10101111        11100110
```

上位4 bit だけを残し，他のbit を 1 にする マスク(mask)

- XOR命令

```
  10100110
xor)00001111
  --------
  10101001
```

下位4 bit だけ各bit の値を反転する マスク(mask)

アセンブリ言語CASL Ⅱ

- 演習 XOR ... #00FF ⊕ #0F0F を計算するプログラムを完成させて実行し，結果を確認せよ

```
XOR     START
        LAD     GR0,#00FF
        [          ]
        RET
DAT     DC  #0F0F
RSLT    DS  1
        END
```

- 排他的論理和に代えて，論理積，即ち #00FF ∧ #0F0F とすると結果がどうなるか確認せよ
- 排他的論理和は [　　]，
  論理積は [　　] の機能がある.

アセンブリ言語CASL Ⅱ

- 演習 AOX ... (#00FF ∧ #0FF3) ∨ #F0FC) ⊕ #FF00)を計算するプログラムを完成させて実行し，途中経過も含めて答えを求めよ

```
AOX     START
        LAD     GR0,#00FF
        [          ]    ; 1 → GR1 ... インデックシング用
        AND     GR0,DAT      ; #00FF ∧ #0FF3 → GR0
        ST      GR0,RSLT     ; <GR0> → RSLT
        [          ]    ; GR0 ∨ #F0FC → GR0
        [          ]    ; <GR0> → RSLT+1
        [          ]    ; 2 → GR1 ... インデックシング用
        XOR     GR0,DAT,GR1  ; GR0 xor #FF00 → GR0
        ST      GR0,RSLT,GR1 ; <GR0> → RSLT+2
        RET
DAT     DC  #0FF3,#F0FC,#FF00
RSLT    DS  3
        END
```


COMET Ⅱマシンの機械命令

(教科書 p.21)

- 算術比較(compare arithmetic)命令

[書き方] CPA　gr,adr[,xr]
[機能] if <GRn> ＞ <e>
then {0→ZF; 0→SF}
else if <GRn> ＝ <e>
then {1→ZF; 0→SF}
else {0→ZF; 1→SF};
0→OF}

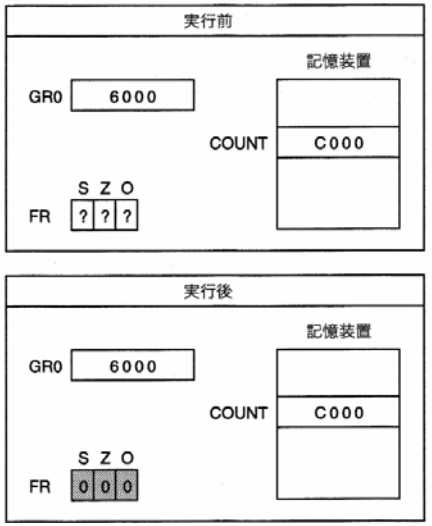
算術減算(SUBA)をしたふりをし，FRだけをセットする命令

ただし，符号付き数同士の比較

[つまり] <GRn>－<e> ＝ t とし，
<t> ＞ 0 ... {0→ZF; 0→SF}
<t> ＝ 0 ... {1→ZF; 0→SF}
<t> ＜ 0 ... {0→ZF; 1→SF}
そして，常に {0→OF}

(1語命令)
[書き方] CPA　gr1,gr2
[機能]　上記と同様

[実行例] CPA　GR0,COUNT



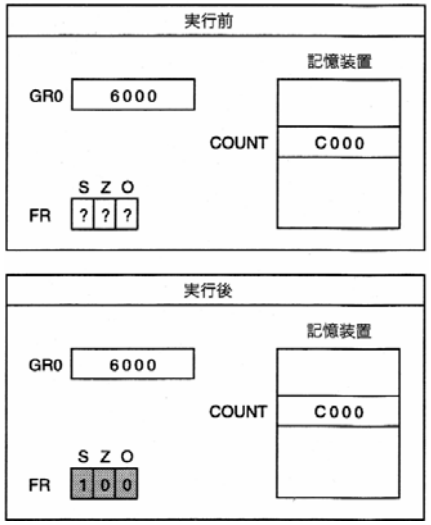
COMET Ⅱマシンの機械命令

（教科書 p.21）

- 論理比較（compare logical）命令

[書き方] CPL gr,adr[,xr]
[機能] if <GRn> ＞ <e>
then {0→ZF; 0→SF}
else if <GRn> ＝ <e>
then {1→ZF; 0→SF}
else {0→ZF; 1→SF};
0→OF}
ただし，符号なし数同士の比較

[実行例] CPL GR0,COUNT

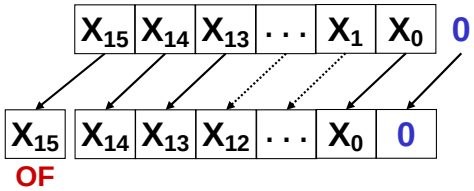


（1語命令）
[書き方] CPL gr1,gr2
[機能] 上記と同様

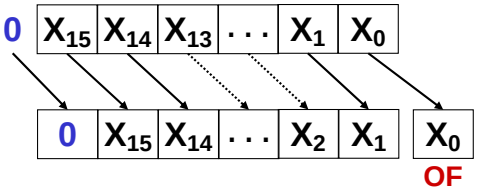
シフト命令

- Logical shift（論理シフト）
  - 最上位ビットを特別扱いしない
  - 左シフト時
    - 空いたビットには 0 をシフト・イン
    - 最後に [15]からシフト・アウトされた値が OF にセットされる
  - 右シフト時
    - 空いたビットには 0 をシフト・イン
    - 最後に [0]からシフト・アウトされた値が OF にセットされる

左1ビット（これをシフト数分繰り返す）



右1ビット（これをシフト数分繰り返す）



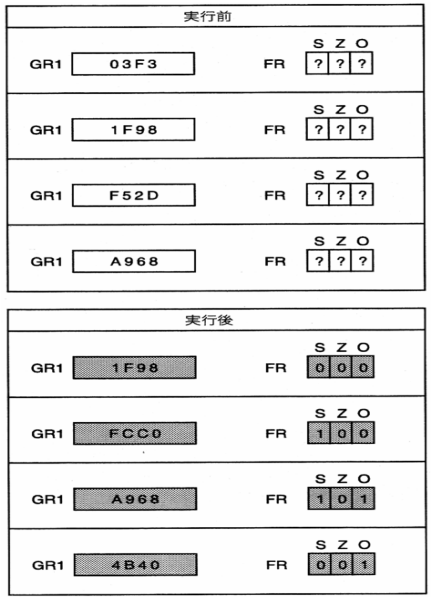
COMET Ⅱマシンの機械命令

（教科書 p.24）

- 論理左シフト
  （shift left logical）命令

[書き方] SLL gr,adr[,xr]
[機能] 0→OF; e → t（シフト数）;
while (t≠0){
<GRn[15]>→OF;
14 → i;
while (i≧0){
<GRn[i]> → GRn[i+1];
i−1 → i
};
0 → GRn[0];
t−1 → t
};
SZの設定

[実行例] SLL GR1,3



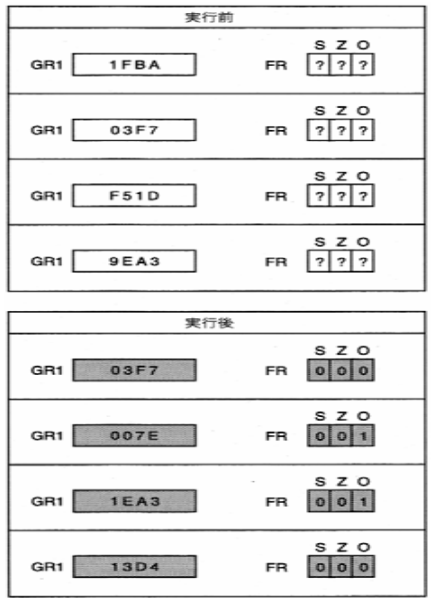
COMET Ⅱマシンの機械命令

（教科書 p.24）

- 論理右シフト
  （shift right logical）命令

[書き方] SRL gr,adr[,xr]
[機能] 0→OF; e → t（シフト数）;
while (t≠0){
<GRn[0]>→OF;
1 → i;
while (i＜16){
<GRn[i]> → GRn[i−1];
i+1 → i
};
0 → GRn[15];
t−1 → t
};
SZの設定

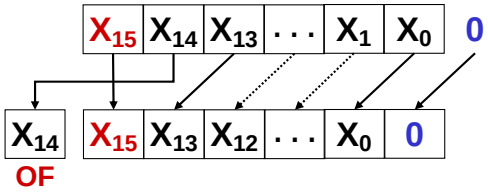
[実行例] SRL GR1,3



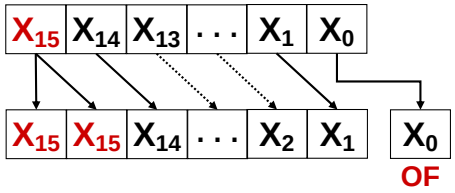
シフト命令

- Arithmetic shift（算術シフト）
  - 最上位ビット（符号ビットと見なす）は変化しない
  - 左シフト時
    - 空いたビットには 0 をシフト・イン
    - 最後に [14]からシフト・アウトされた値が OF にセットされる
  - 右シフト時
    - 空いたビットには 符号ビット をシフト・イン
    - 最後に [0]からシフト・アウトされた値が OF にセットされる

左1ビット（これをシフト数分繰り返す）



右1ビット（これをシフト数分繰り返す）



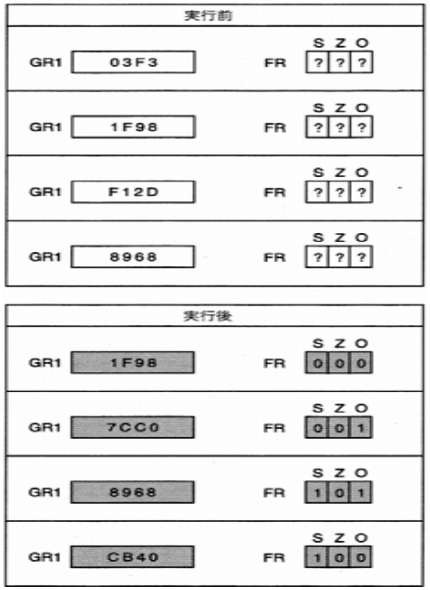
COMET Ⅱマシンの機械命令 （教科書 p.22）

- 算術左シフト
  （shift left arithmetic）命令
  [書き方] SLA gr,adr[,xr]
  [機能] 0→OF; e → t（シフト数）;

```
while (t≠0){
  <GRn[14]>→OF;
  13 → i;
  while (i≧0){
    <GRn[ i ]> → GRn[ i+1 ];
    i−1 → i
  };
  0→ GRn[0];
  t−1 → t
};
```

  SZの設定

[実行例] SLA GR1,3



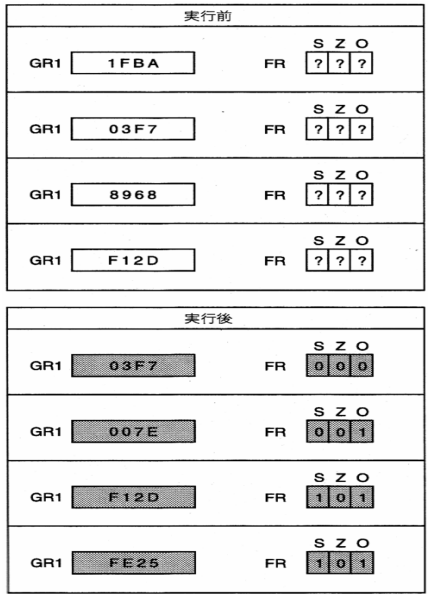
COMET Ⅱマシンの機械命令 （教科書 p.22）

- 算術右シフト
  （shift right arithmetic）命令
  [書き方] SRA gr,adr[,xr]
  [機能] 0→OF; e → t（シフト数）;

```
while (t≠0){
  <GRn[0]>→OF;
  1 → i;
  while (i＜16){
    <GRn[ i ]> → GRn[ i−1 ];
    i+1 → i
  };
  t−1 → t
};
```

  SZの設定

[実行例] SRA GR1,3



アセンブリ言語CASLⅡ

- 演習 SHFT ... −1 を1ビット算術左シフトした数（−2）と，−1 を3ビット算術左シフトした数（−8）を加算し，0から算術減算すると 10 になることをプログラムを書いて確かめよ

```
SHFT    START
        LAD     GR1,-1
        ;
        LAD     GR2,-1,
        ;
        LAD     GR0,0
        ;
        ST      GR0,RSLT
        RET
RSLT    DS      1
        END
```

- 演習 SHFT2 ... 上記と同じ事を，論理シフト，論理減算するとどうなるかを確かめよ

アセンブリ言語CASLⅡ

- 演習 SHFT2 … −1 を1ビット論理左シフトした数(−2)と，−1を3ビット論理左シフトした数(−8)を加算し，0から論理減算するプログラムを書いて，結果を確かめよ

```
SHFT2   START
        LAD     GR1,-1
                ;
        LAD     GR2,-1
                ;
                ;
        LAD     GR0,0
                ;
        ST      GR0,RSLT
        RET
RSLT    DS  1
        END
```


宿題：

本日配布した「第5回宿題」を最後まで完成させ，次回授業の冒頭に副手に提出すること