プログラミング演習１
課題4：アセンブリ言語

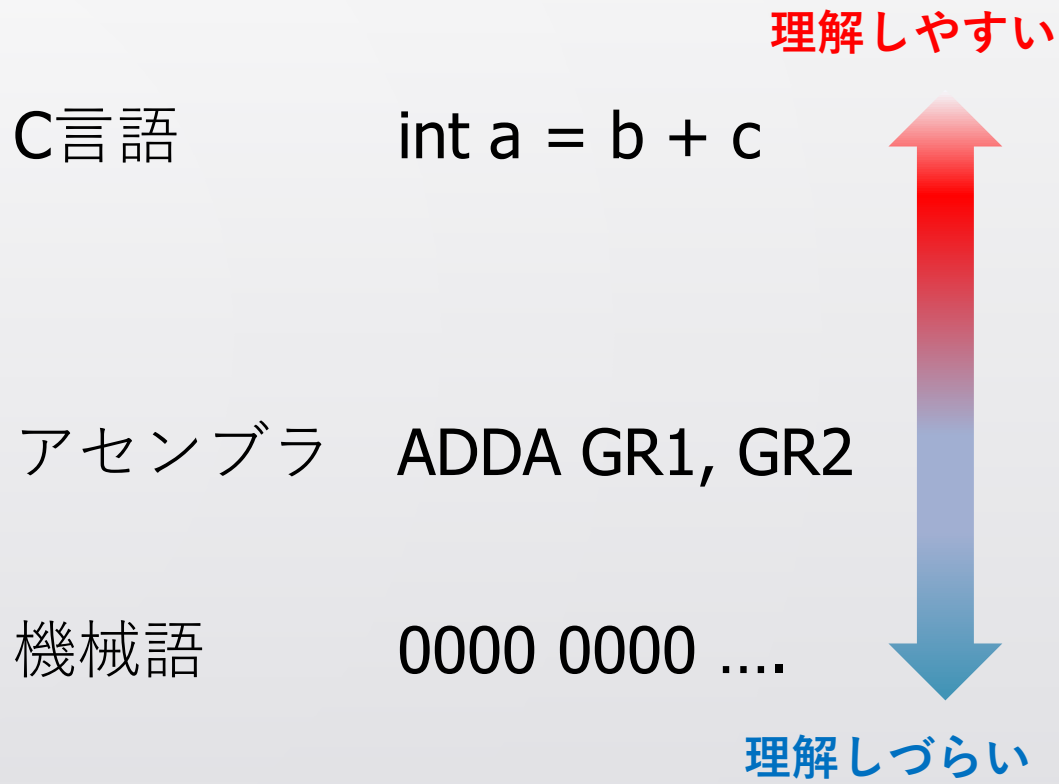
入出力と条件分岐

本日の授業内容

1. アセンブラって何？
2. 例題１：短い文を印刷する
3. 例題２：大小判定
4. 例題３：読み書きの例題
5. 練習問題

アセンブラって何？

- プログラミング言語の分類
  - 高水準言語（C言語，Javaなど）
    - 人間にとってわかりやすい
  - 低水準言語（アセンブラ言語・機械語）
    - 機械にとって理解しやすい
  - アセンブラ言語
    - 機械語を記号で書いたもの
    - 人間にとって（機械語よりも）理解しやすい



CASL II

- 基本情報技術者試験のために開発されたアセンブリ言語
- 仮想計算機“COMET II”で実行できる

汎用レジスタ(各16bit)

GR0
GR1
GR2
GR3
GR4
GR5
GR6
GR7

プログラムレジスタ（16bit）

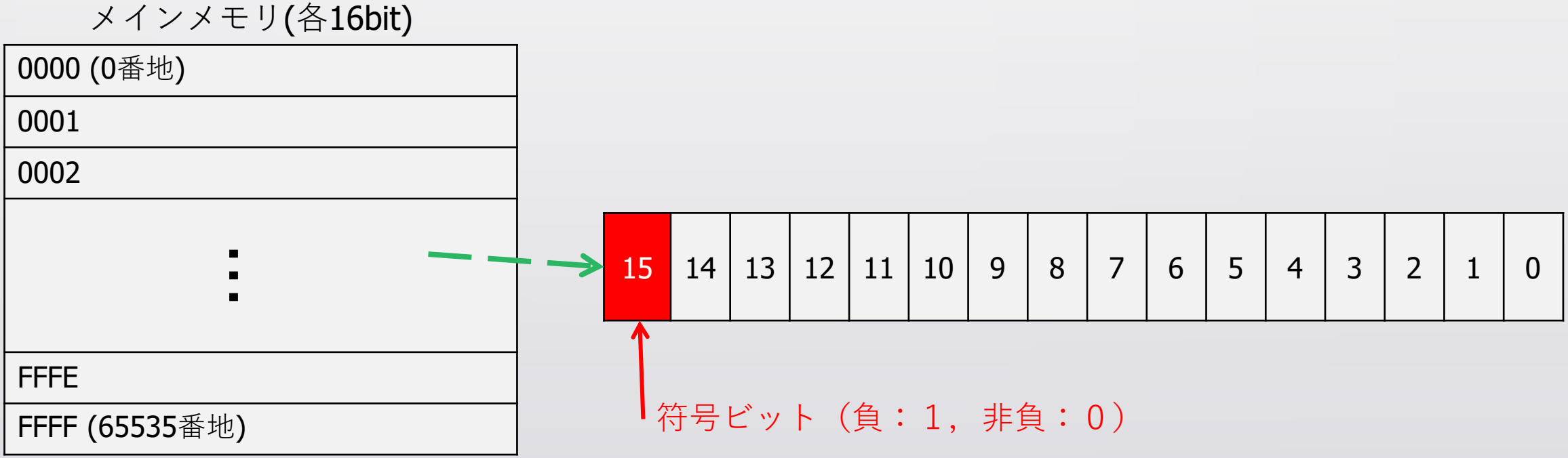
スタックポインタ（16bit）

フラグレジスタ(3bit)

overflow flag (1bit)	sign flag (1bit)	zero flag (1bit)

CASL II

- 基本情報技術者試験のために開発されたアセンブリ言語
- 仮想計算機“COMET II”で実行できる



例題説明の前に・・・

- レポート課題の内容
  - CASL2による数値の判定
    1. 素数かどうか判定
    2. 16の倍数かどうか判定
    3. 8の倍数かどうか判定
    4. 4の倍数かどうか判定
    5. 2の倍数かどうか判定

例題１：短い文を印刷する

- アセンブリ言語の書き方

```
;-----------------------------
; 例1　文字列の出力
;-----------------------------
EX1     START
        OUT     OUTBUF,OUTLEN   ; 文字を出力する
        RET
;
OUTBUF  DC      'HELLO WORLD!'  ;  ←適当に変えてみる
OUTLEN  DC      12              ;  ←OUTBUFに合わせて変える
        END
```

ラベル欄　　命令コード欄　　オペランド欄　　注釈 (コメント) 欄

例題１：短い文を印刷する

- 命令コード欄について（講義資料の命令コード一覧を参照）
  - START命令で始まり
  - END命令で終了

```
;-------------------------------
; 例1　文字列の出力
;-------------------------------
EX1     START
        OUT     OUTBUF,OUTLEN   ; 文字を出力する
        RET
OUTBUF  DC      'HELLO WORLD!'  ;  ←適当に変えてみる
OUTLEN  DC      12              ;  ←OUTBUFに合わせて変える
        END
```

命令コード欄

例題１：短い文を印刷する

- 命令コード欄について（講義資料の命令コード一覧を参照）
  - 非実行命令
    - DC (Define Constant): 定数配置
    - DS (Define Storage): 領域の確保

```
;-----------------------------
; 例1　文字列の出力
;-----------------------------
EX1     START
        OUT     OUTBUF,OUTLEN   ; 文字を出力する
        RET
OUTBUF  DC      'HELLO WORLD!'  ;  ←適当に変えてみる
OUTLEN  DC      12              ;  ←OUTBUFに合わせて変える
        END
```

命令コード欄

例題１：短い文を印刷する

- オペランド欄について
  - 各命令に使うレジスタやアドレスを記入
    (命令コードの後に1文字以上の空白を置く)
  - 命令によって必要な場合と不必要な場合がある
  - 次の行に継続して書くことはできない

```
;-----------------------------
; 例1　文字列の出力
;-----------------------------
EX1     START
        OUT     OUTBUF,OUTLEN   ; 文字を出力する
        RET
OUTBUF  DC      'HELLO WORLD!'  ;  ←適当に変えてみる
OUTLEN  DC      12              ;  ←OUTBUFに合わせて変える
        END
```

オペランド欄

例題１：短い文を印刷する

- コメント欄について
  - セミコロン以降の文字を注釈として扱う
  - オペランド欄に文字列として含まれるセミコロンは例外

```
;-----------------------------
; 例1　文字列の出力
;-----------------------------
EX1     START
        OUT     OUTBUF,OUTLEN
        RET
OUTBUF  DC      'HELLO WORLD!'
OUTLEN  DC      12
        END
```

コメント欄

```
; 文字を出力する

;   ←適当に変えてみる
;   ←OUTBUFに合わせて変える
```

例題１：短い文を印刷する

OUT命令

- 構文：OUT　　ラベル１,ラベル２
  - ラベル１：出力領域
  - ラベル２：出力文字長が入る1語長の領域

```
;-----------------------------
; 例1　文字列の出力
;-----------------------------
EX1     START
        OUT     OUTBUF,OUTLEN   ; 文字を出力する
        RET
OUTBUF  DC      'HELLO WORLD!'  ;  ←適当に変えてみる
OUTLEN  DC      12              ;  ←OUTBUFに合わせて変える
        END
```

- OUT　　OUTBUF,OUTLEN　　;文字を出力する
  - OUTBUF：出力する文字列が格納されているメモリ領域のラベル名
  - OUTLEN: 出力する文字列の長さが格納されている領域のラベル名（1語長）

OUTBUF，OUTLENを変更して実行してみましょう

例題２:大小判定

- 説明する命令
  1. LD命令
  2. DC命令
  3. CPA命令
  4. 分岐命令
     （JMI, JZEなど）

```
;-------------------------
; 例2　比較と条件分岐
;-------------------------
EX2     START
        LD      GR1,DATAA       ;GR1 ← ( DATAA )
        CPA     GR1,DATAB       ;( GR1 ) － ( DATAB )
        JMI     LESS            ;FRの値が X1X の時に LESS へ分岐する
        JZE     EQUAL           ;FRの値が XX1 の時に EQUAL へ分岐する
        OUT     OTMSG1,OUTLEN   ;文字を出力する
        JUMP    OWARI           ;無条件に OWARI へ分岐する
EQUAL   OUT     OTMSG2,OUTLEN   ;文字を出力する
        JUMP    OWARI           ;無条件に OWARI へ分岐する
LESS    OUT     OTMSG3,OUTLEN   ;文字を出力する
OWARI   RET
DATAA   DC      3
DATAB   DC      8
OTMSG1  DC      'B is less than A'
OTMSG2  DC      'B is equal to A '
OTMSG3  DC      'A is less than B'
OUTLEN  DC      16
        END
```

例題２:大小判定

- LD命令
  - 構文1

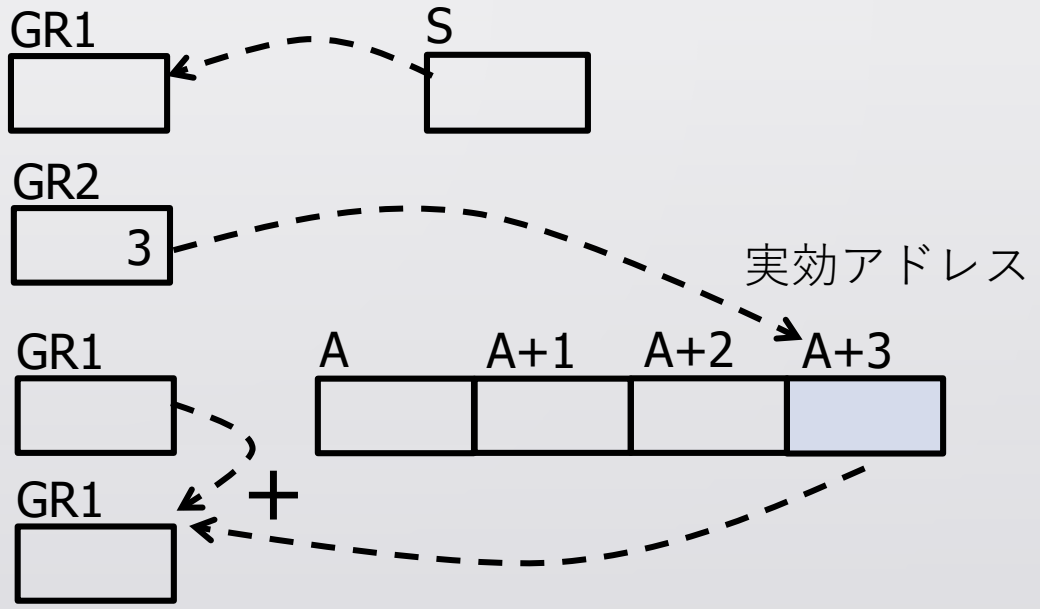
    [ラベル] LD r1,r2・・・レジスタr2の内容をレジスタr1にロードする
  - 構文2

    [ラベル]LD r1,<実効アドレス>・・・実効アドレスの内容をr1にロードする
- 汎用レジスタ
  - GR0からGR7まで存在（予約語なのでラベル・変数には使えない）
  - GR0以外は指標レジスタ

指標レジスタ（インデックスレジスタ）

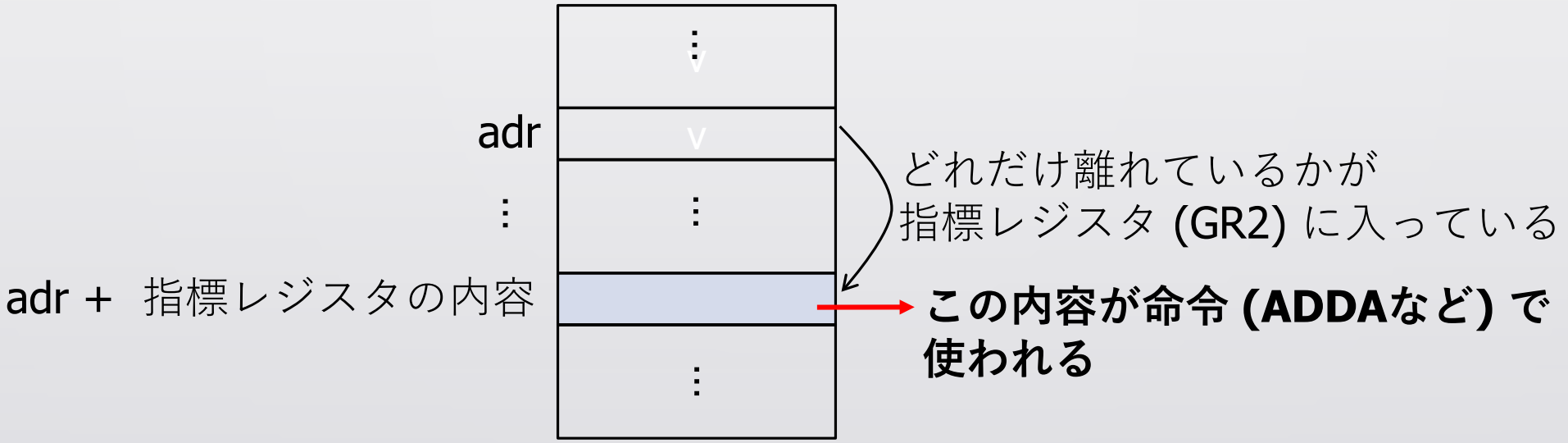
- アクセスしたいメモリ上の番地（アドレス）の基準値からの相対的な値を格納
  - 例）データAのアドレスから3番目の内容を加算（GR2が指標レジスタ）

```
LD      GR1, S
LAD     GR2, 3
ADDA    GR1, A, GR2
ST      GR1,S
```



実効アドレス

- 処理の対象となるアドレスそのもの
  - 例）ADDA GR1, adr, GR2 ⇒ adr+(GR2の内容)



DC命令

- 定数宣言
  - 10進定数

    [ラベル]　DC　3
  - 16進定数

    [ラベル]　DC　#00C7  $(0000\ 0000\ 1100\ 0111)_2$
  - 文字定数

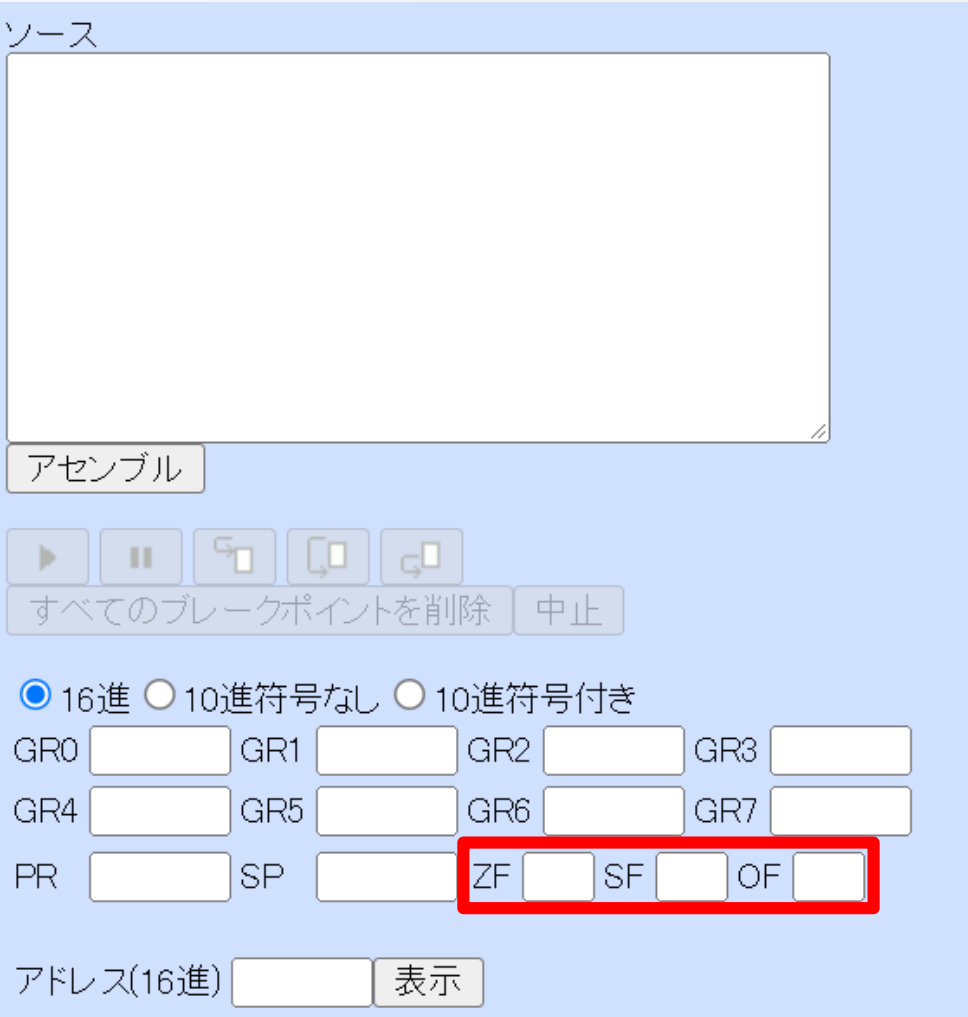
    [ラベル]　DC　‘文字列’
  - アドレス定数

    [ラベル]　DC　ラベル名 (DATAAなど)

CPA(ComPare Arithmetic, 算術比較)命令

フラグレジスタ

Flag	Bit	ビットが設定される条件
OF (Overflow Flag)	1	・算術演算の結果が-32768～32767に収まらなくなったとき ・論理演算の結果のが0～65535に収まらなくなったとき ・シフト演算でレジスタから最後に送り出されたビットが格納される
	0	上記以外のとき
SF (Sign Flag)	1	演算結果の符号が負 (MSBが1) のとき
	0	演算結果の符号が正 (MSBが0) のとき
ZF (Zero Flag)	1	演算結果が零 (全部のビットが0) のとき
	0	上記以外のとき



CPA(ComPare Arithmetic, 算術比較)命令

- 構文
  - [ラベル] CPA r1, r2
    - レジスタr1の内容とレジスタr2の内容を符号付き数として比較
  - [ラベル] CPA r, adr[,x]
    - レジスタrの内容とadr[,x](実効アドレス)が指すメモリの内容を符号付き数として比較
  - CPA命令の結果(CPA r1, r2の場合)

比較結果	r1 > r2	r1 = r2	r1 < r2
SF	0	0	1
ZF	0	1	0

分岐命令

- フラグレジスタの値（演算結果）で分岐先を決定
  - JPL (Jump on PLus) : SFとZFが両方とも0のとき分岐
  - JMI (Jump on MInus) : SFが1のとき分岐
  - JNZ (Jump on Non Zero) : ZFが0のとき分岐
  - JZE (Jump on ZEro) : ZFが1のとき分岐
  - JOV (Jump on OVerflow) : OFが1のとき分岐
  - JUMP : 無条件に分岐

例題３：読み書きの例題

- 説明する命令
  - IN命令
  - ループ

```
;-----------------------------
; 例3　ループ
;-----------------------------
EX3     START
YOMU    IN      INBUF,INLENG    ;文字を入力する
        LD      GR1,INLENG      ;GR1 ← ( INLENG )
        JZE     OWARI           ;FRの値が XX1 の時に OWARI へ分岐する
        OUT     INBUF,INLENG    ;文字を出力する
        JUMP    YOMU            ;無条件に YOMU へ分岐する
OWARI   RET
INBUF   DS      80
INLENG  DS      1
        END
```

IN命令

- 構文
  - [ラベル] IN ラベル1, ラベル2
    - ラベル1: 入力領域を指すラベル
    - ラベル2: 入力文字長が入る1語長の領域を指すラベル
  - C言語のscanfのような命令

ループ

- フラグレジスタ・分岐命令を応用して繰り返し処理（ループ）を実現

```
;-----------------------------
; 例3　ループ
;-----------------------------
EX3     START
YOMU    IN      INBUF,INLENG    ;文字を入力する
        LD      GR1,INLENG      ;GR1 ← ( INLENG )
        JZE     OWARI           ;FRの値が XX1 の時に OWARI へ分岐する
        OUT     INBUF,INLENG    ;文字を出力する
        JUMP    YOMU            ;無条件に YOMU へ分岐する
OWARI   RET
INBUF   DS      80
INLENG  DS      1
        END
```

終了条件
LD命令の結果ZF=1となる
→ロードする文字がない

練習課題：数値の判定

- 手順
  1. マスク処理について調査せよ
  2. データ領域にあらかじめ記憶されている10進数データが奇数か偶数か判定
  3. 4の倍数，8の倍数の判定も加える

- 課題が完成したら
  - 16の倍数にも挑戦してみましょう