プログラミング演習1
課題4:アセンブリ言語

第1回 入出力と条件分岐

- 座席は自由です
- 講義資料ページを開いておく
- WebClassで出席を登録する準備

授業の進め方

- 基本的に例題+課題 (宿題)
- 課題の進捗状況は毎回TAがチェック

課題がすべてクリアできた状態でないと
レポート課題に取り組むのはほぼ無理です

- 各課題のチェック受付は授業中 or 翌週1回のみ

授業の進め方

第1回目の課題は
第2回目までに終わらせる

	授業		自宅 or 空き時間
第1回目	例題を使った説明 (約60分)	課題+チェック (約30分)	各課題への取り組み
第2回目	例題を使った説明 (約60分)	課題+チェック (約30分)	各課題への取り組み
第3回目	例題を使った説明 (約60分)	課題+チェック (約30分)	各課題への取り組み

第1回目，2回目の
課題のみチェック受付

第2回目の課題は
第3回目までに終わらせる

第2回目，3回目の
課題のみチェック受付

第1回目の課題は
受付けない

授業の進め方

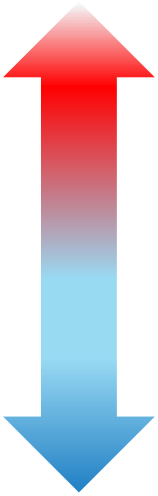
課題がすべてクリアできた状態でないと
レポート課題に取り組むのはほぼ無理です

- 第3回目の時点で課題の進捗が遅い人
  ⇒ レポートで追加課題等の措置を取る
- 分からないことはなるべく授業中に解決を！
- 遅刻，欠席の場合は必ず事前連絡
  - 病欠の場合は領収書等の証明書を提出

アセンブラってなに？

- C言語，C++等 (高水準言語，上級言語)

自然語に近い
(printf, getなど)



人間にとってかなり理解しやすい

機械語に近い
(2進数，16進数を用いる)

人間にとってやや理解しやすい

- アセンブラ言語 (低水準言語，下級言語)

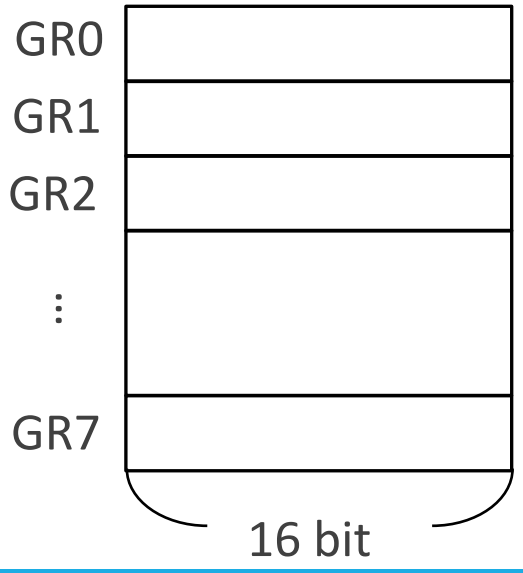
機械語　0010 0000 0001 0000 0000 0000 0000 1010

アセンブラ言語　ADDA GR1, ADDRESS

機械語を記号で書いたもの

CASL IIについて

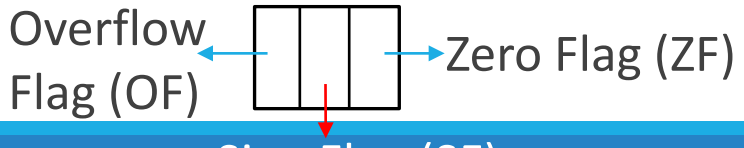
- 基本情報技術者試験で出題される言語
- 架空のハードウエアCOMET IIで実行可能なアセンブラ言語として定義
  - 汎用レジスタ



  - プログラムレジスタ

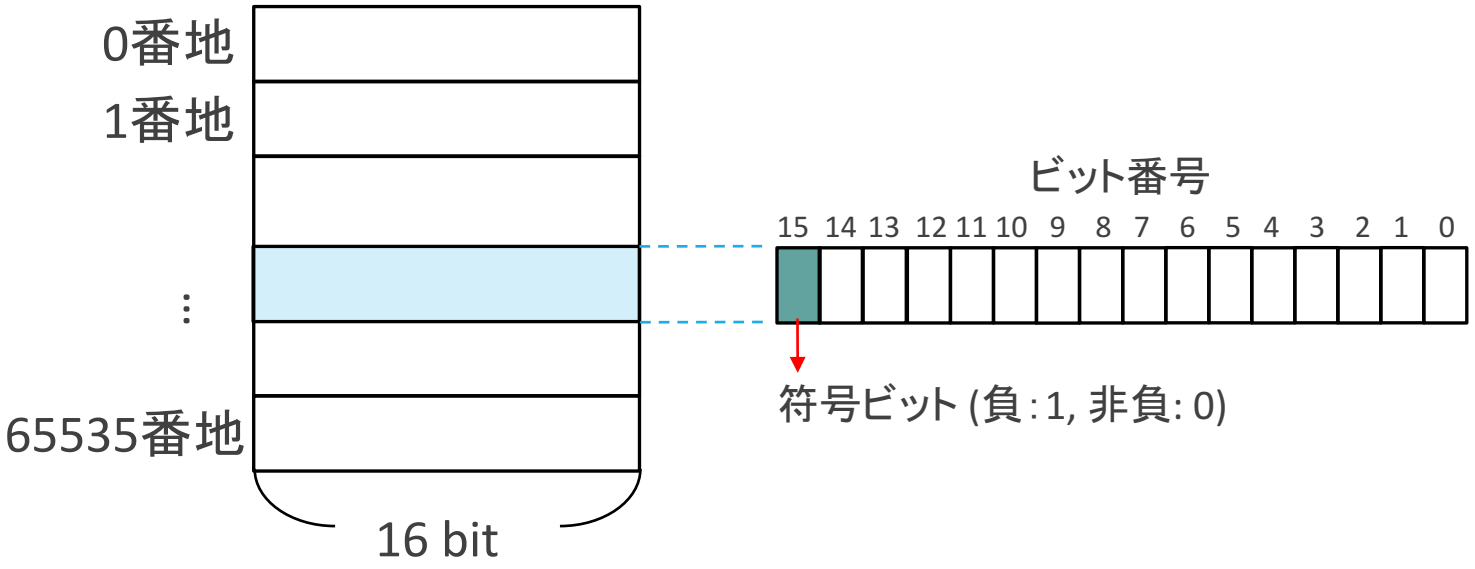
  - スタックポインタ

  - フラグレジスタ 3 bit



CASL IIについて

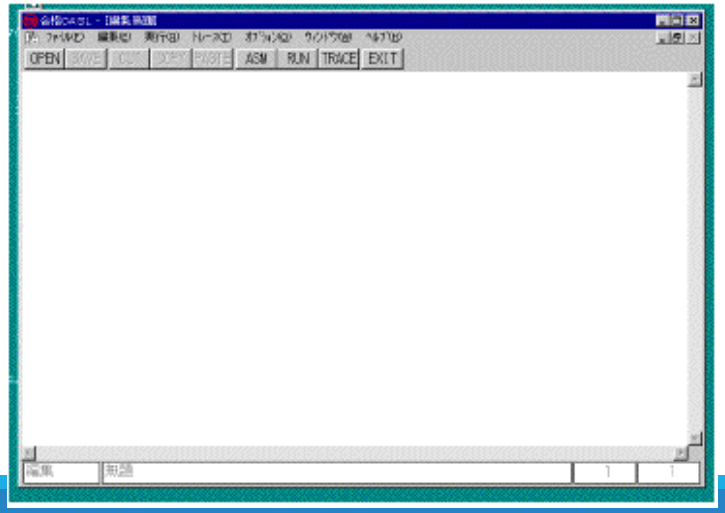
- 基本情報技術者試験で出題される言語
- 架空のハードウエアCOMET IIで実行可能なアセンブラ言語として定義
  - 主記憶



CASL IIについて

- この授業で使うアプリケーションソフト：合格CASL II
  - CASL II アセンブラ
  - CASL II シミュレータ
  - CASL II デバッガ

  ⇒ 起動，使用方法は講義ページを参照



例題1:短い文を印刷する

- アセンブリ言語の書き方

例題1

```
;-----------------------------
; 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

- アセンブリ言語の書き方

例題1

```
;-----------------------------
; 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

- ラベル名は8文字以内
- 先頭の文字は大文字のアルファベット
  (2文字目以降は大文字のアルファベット、数字のいずれでも可)
- 1文字目が空白の場合は命令コードと解釈される

例題1:短い文を印刷する

- アセンブリ言語の書き方

例題1

```
;-----------------------------
; 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

講義ページの「命令コード」一覧を参照
http://www.ced.is.utsunomiyau.ac.jp/lecture/2019/prog/p1/kadai4/spec.htm

例題1:短い文を印刷する

- アセンブリ言語の書き方

例題1

```
;-----------------------------
; 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
```

講義ページの「命令コード」一覧を参照
http://www.ced.is.utsunomiyau.ac.jp/lecture/2019/prog/p1/kadai4/spec.htm

例題1:短い文を印刷する

- アセンブリ言語の書き方

例題1

```
;-----------------------------
; 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
```

- 非実行命令
  - DC (Define Constant): アドレス定数
  - DS (Define Storage): 領域の確保

    ⇒プログラムの後半にまとめて記述する

例題1:短い文を印刷する

- アセンブリ言語の書き方

例題1

```
;----------------------------
; 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

- 各命令に使うレジスタやアドレスを記入する
  (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

例題1:短い文を印刷する

- アセンブリ言語の書き方

例題1

```
;-----------------------------
; 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
```

注釈 (コメント) 欄

- 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

例題1:短い文を印刷する

- 命令コードの説明

例題1

```
;-----------------------------
; 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
```

例題1:短い文を印刷する

- マクロ命令 (IN, OUT, RPUSH, RPOP)
  - 入力命令 IN
    [ラベル] IN ラベル1, ラベル2

入力領域をさすラベル名

入力文字長が入る1語長の
領域をさすラベル名

  - 出力命令　OUT
    [ラベル] OUT ラベル1, ラベル2

出力領域をさすラベル名

出力文字長が入る1語長の
領域をさすラベル名

例題1:短い文を印刷する

- 命令コードの説明

例題1

```
;-----------------------------
; 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
```

出力領域をさすラベル名: OUTBUF, 中身は文字列　HELLO WORLD!

出力文字長をさすラベル名: OUTLEN, 12 bit

☆OUTBUF, OUTLEN を自由に変更して実行してみましょう。

例題2:大小判定

```
;-----------------------------
; 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

```
;----------------------------
; 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 (LoaD)　命令とレジスタ

  [ラベル]　LD r1, r2

  ⇒ r2で指定したレジスタの内容をr1で指定したレジスタに入れる

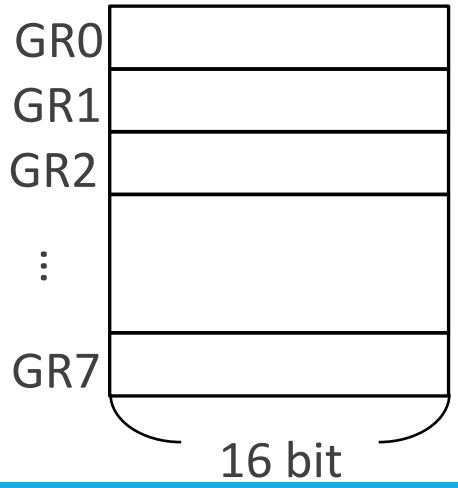
  [ラベル]　LD r, adr[,x]

  ⇒ adr[,x] で指定した主記憶の内容をrで指定したレジスタに入れる

  「実効アドレス」

例題2:大小判定

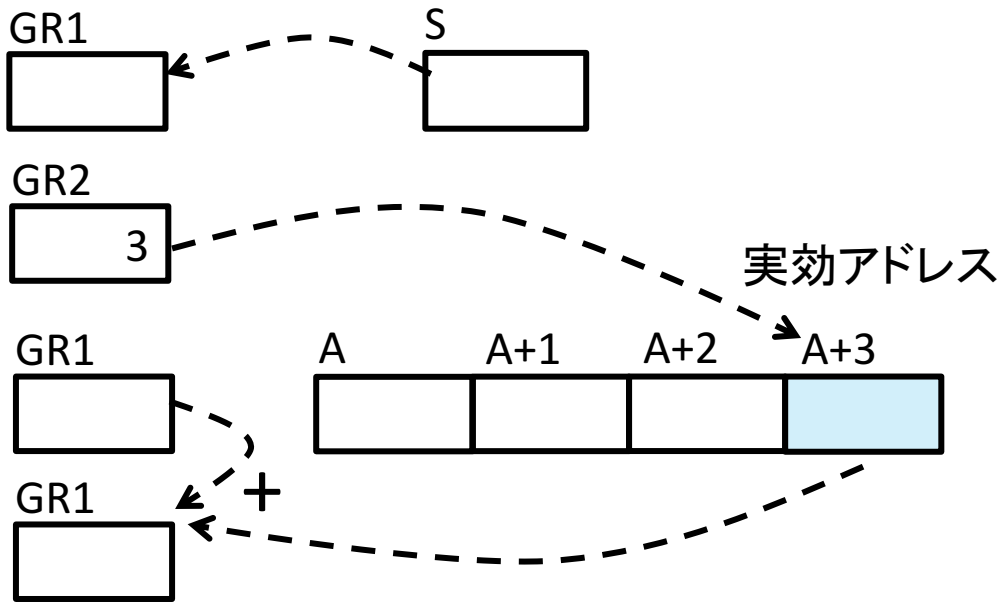
- 汎用レジスタ
  - GR0からGR7まで存在
  - 予約語なので他の変数としては使えない
  - GR1からGR7は指標レジスタとしても使用



例題2:大小判定

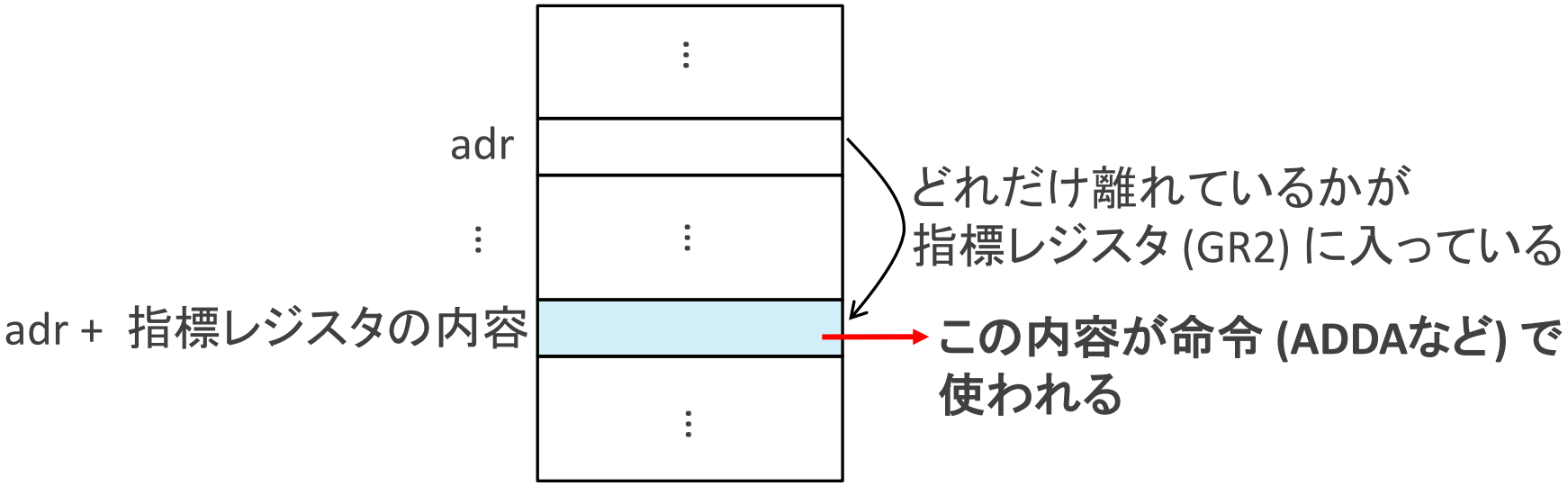
- 指標レジスタ (汎用レジスタの使い方の1方法)
  - 例 : データAのアドレスから3番目の内容を加算する

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



例題2:大小判定

- 実効アドレス
  - 指示したアドレスに指標レジスタの内容を加算した値
  - 先ほどの例の場合 : ADDA GR1, A, GR2 ⇒ A+3



例題2:大小判定

```
;----------------------------
; 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
```

DATAAという名前のついた主記憶に
3 (10進数) を入れる

例題2:大小判定

- 定数の記述方法 (DC命令; Define Constant)
  - 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)

例題2:大小判定

```
;----------------------------
; 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

- CPA (ComPare Arithmetic;算術比較) 命令

  [ラベル]　CPA r1, r2

  ⇒ r1で指定したレジスタの内容と
  　r2で指定したレジスタの内容を符号付きの数として比較

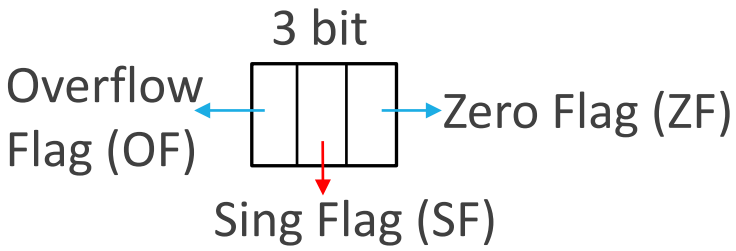
  [ラベル]　CPA r, adr[,x]

  ⇒ rで指定した内容と adr[,x] (実効アドレス) で指定した
  　主記憶の内容を符号付きの数として比較

結果によってFR (Flag Register) の値が設定される

例題2:大小判定

- フラグレジスタ
  - OF (Overflow Flag)
  - SF (Sign Flag)
  - ZF (Zero Flag)



例題2:大小判定

- 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 (ビット番号15が1) のとき
	0	演算結果の符号が正 (ビット番号15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

例題2:大小判定

- CPA (ComPare Arithmetic;算術比較) 命令

  [ラベル]　CPA r1, r2

  ⇒ r1で指定したレジスタの内容と
  　r2で指定したレジスタの内容を符号付きの数として比較

  [ラベル]　CPA r, adr[,x]

  ⇒ rで指定した内容と adr[,x] (実効アドレス) で指定した
  　主記憶の内容を符号付きの数として比較

比較結果	GR1 > B	GR1 = B	GR1 < B
SF	0	0	1
ZF	0	1	0

例題2:大小判定

```
;-----------------------------
; 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
```

DATABという名前の主記憶には
8 (10進数) が入っている

例題2:大小判定

```
;-----------------------------
; 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

- 分岐命令 : 演算結果とフラグレジスタの値で分岐先を決める
  - 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2:大小判定

```
;----------------------------
; 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3: 読み書きの例題

```
;----------------------------
; 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
```

例題1:短い文を印刷する

- マクロ命令 (IN, OUT, RPUSH, RPOP)
  - 入力命令 IN

    [ラベル] IN ラベル1, ラベル2

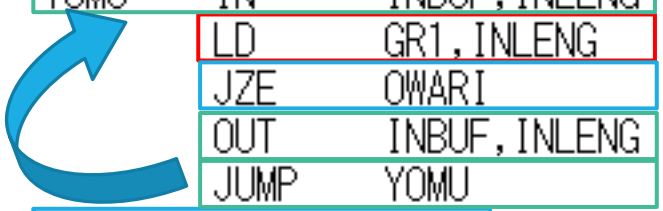
入力領域をさすラベル名
(入力された内容)

入力文字長が入る1語長の領域をさすラベル名
(入力された内容の長さを格納)

例題3: 読み書きの例題

```
;----------------------------
; 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
```

例題3: 読み書きの例題



```
;---------------------------
; 例3　ループ
;---------------------------
EX3     START
YOMU    IN      INBUF,INLENG
        LD      GR1,INLENG
        JZE     OWARI
        OUT     INBUF,INLENG
        JUMP    YOMU
OWARI   RET
INBUF   DS      80
INLENG  DS      1
        END
```

```
;文字を入力する
;GR1 ← ( INLENG )
;FRの値が XX1 の時に OWARI へ分岐する
;文字を出力する
;無条件に YOMU へ分岐する
```

課題

☆課題毎にTAのチェックを受けること
☆演習時間内に終わらなかった課題は「宿題」
　⇒ 来週の課題取組時にTAがチェック
☆課題1～4のチェック受付は来週まで

- 課題1
  - マスク処理が必要 ⇒ 要調査
  - データが0や負の場合も考慮
- 課題2
  - データが0や負の場合も「DATA ERROR」と印刷
- 課題3
  - データが0や負の場合も判定
- 課題4
  - マスク処理が必要
  - 課題1～3と同様に0や負の場合も判定を行う