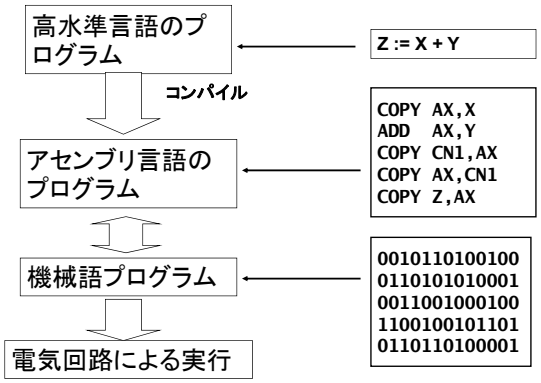
# p88アセンブラ

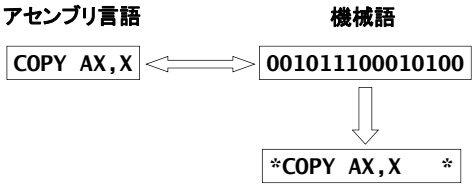
## p88アセンブラとは

- p88アセンブラは、Alan W. Biermann の “Great Ideas in Computer Science” (The MIT Press, 1990, 1997), 邦訳「やさしいコンピュータ科学」(アスキー出版局, 1993)の中でコンピュータ科学の入門学習用に記述されたもの
- この書籍では、p88アセンブラを使って、アセンブリ言語、コン ピュータアーキテクチャ、コンパイラなどの話題を平易に解説している。

## プログラムが実行されるまで

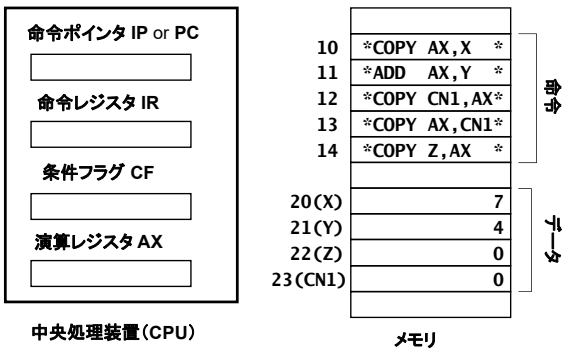



## アセンブリ言語と機械語




アセンブリ言語の命令文と機械語命令文は1対1に対応する.

## コンピュータアーキテクチャ



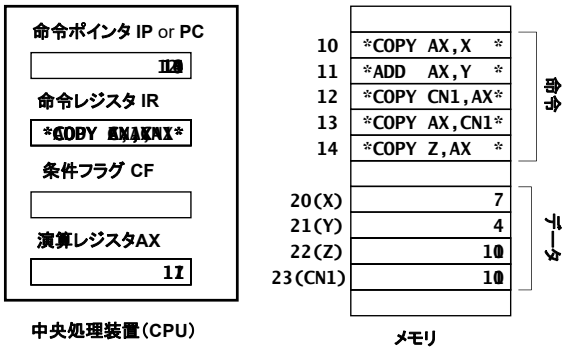

## 命令取り出し-実行サイクル

コンピュータは，次の2つのステップを繰返し実行する.

1. （命令取り出し） PCが指定するアドレスに記憶されている命令を取り出し，その命令をIRに入れる. PCの値を1つ増やす.
2. （命令実行） IR にある命令を実行する.

この2つのステップを，fetch-execution cycle 命令取り出し-実行サイクルという.

# 命令取り出し-実行サイクルの例




# p88計算機の12種の命令(1)

| 書式 | 動作 |
|---|---|
| COPY AX,mem | mem番地の内容をAXへコピー |
| COPY mem,AX | AXの内容をmem番地へコピー |

# p88計算機の12種の命令(2)

| 書式 | 動作 |
|---|---|
| ADD AX,mem | AX:= AX + mem番地の内容 |
| SUB AX,mem | AX:= AX - mem番地の内容 |
| MUL AX,mem | AX:= AX * mem番地の内容 |
| DIV AX,mem | AX:= AX / mem番地の内容 |

# p88計算機の12種の命令(3)

| 書式 | 動作 |
|---|---|
| CMP AX,mem | CF:= AX - mem番地の内容 |

# p88計算機の12種の命令(4)

| 書式 | 動作 |
|---|---|
| JMP lab1 | IP:= lab1 |
| JNB lab1 | if CF >= 0 then IP := lab1 |
| JB lab1 | if CF < 0 then IP := lab1 |

# p88計算機の12種の命令(5)

| 書式 | 動作 |
|---|---|
| IN AX | 整数を読み込んでAXへ代入する |
| OUT AX | AXの内容を出力する |

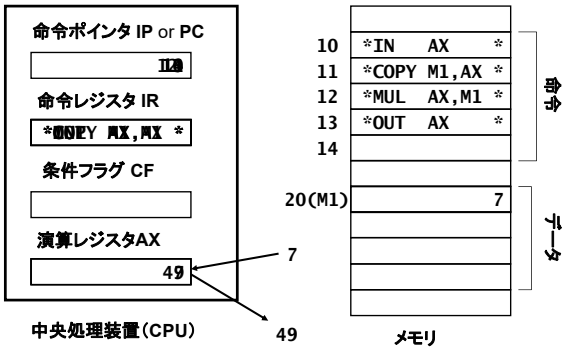
# 簡単なp88プログラム（1）

```
IN  AX
OUT AX
```

# 簡単なp88プログラム（2）

```
IN   AX
COPY M1,AX
MUL  AX,M1
OUT  AX
```

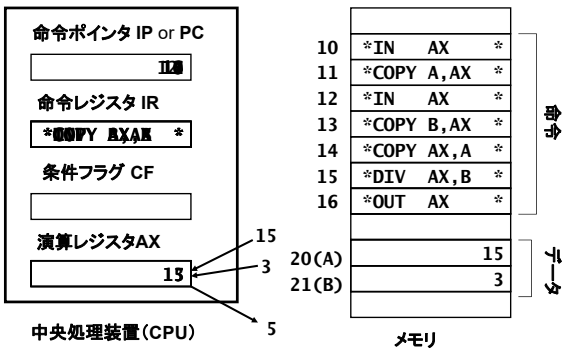
# 簡単なp88プログラム（2）




# 簡単なp88プログラム（3）

```
IN   AX
COPY A,AX
IN   AX
COPY B,AX
COPY AX,A
DIV  AX,B
OUT  AX
```

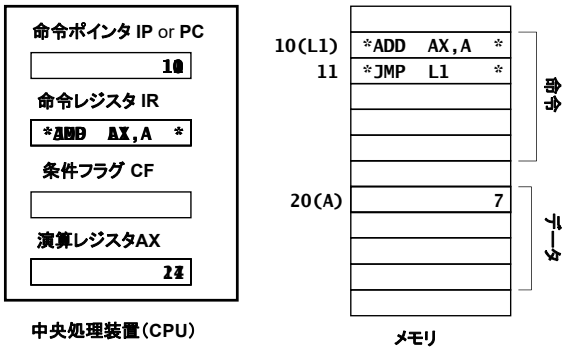
# 簡単なp88プログラム（3）




# ジャンプ命令（1）

```
L1   ADD  AX,A
     JMP  L1
```

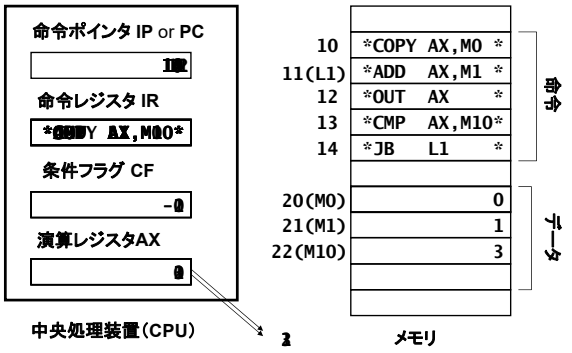
# ジャンプ命令（1）



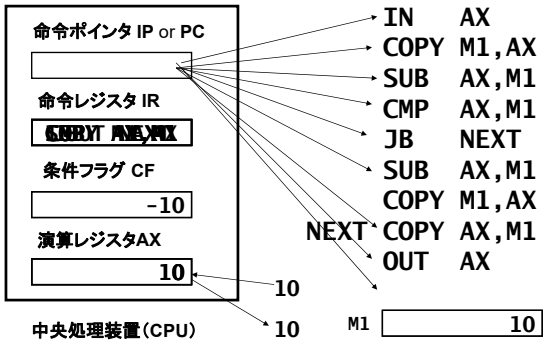

# ジャンプ命令（2）

```
      COPY AX,M0
L1    ADD  AX,M1
      OUT  AX
      CMP  AX,M10
      JB   L1
```

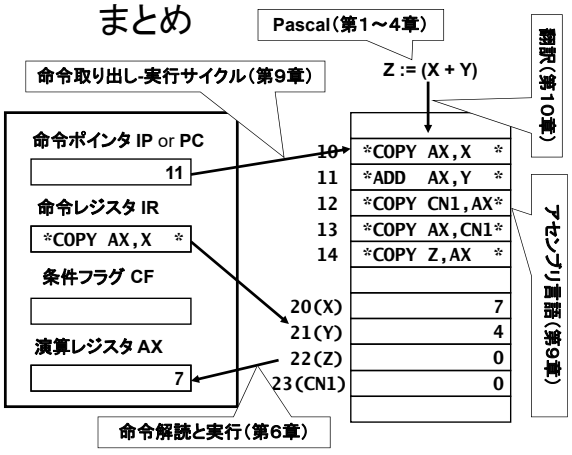
# ジャンプ命令（2）




# ジャンプ命令（3）




# まとめ




# p88 サンプルプログラム

- まず処理系（とシミュレータ）が必要
- 幸いなことに、荻原剛志先生が作成されている。
  - http://homepage.mac.com/t_ogihara/software/CLI/p88/indexj.html
    （現在リンク切れ）

```
LOOP  IN     AX
      ADD    AX,SUM
      COPY   SUM,AX
      COPY   BX,TIMES
      SUB    BX,ONE
      COPY   TIMES,BX
      CMP    BX,ONE
      JNB    LOOP
      COPY   AX,SUM
      OUT    AX
      END
ONE   1
TIMES 5
SUM   0
```

A01.p88

```
;; Make sum of a sequence of numbers.
        copy    ax,#0       immediate
L1
        copy    bx,data+c(ax)     Indirect displacement
;;              negative means end of data
        cmp     bx,#0
        jb      L2
        add     bx,sum
        copy    sum,bx
;;              each data is 2 byte long
        add     ax,#2
        jmp     L1
L2      copy    bx,sum
        out     bx
        end
sum     0
data    1
        2
        3
        4
        5
        -1
```

A02.p88

```
;; Sum of a sequence of numbers.
        copy    ax,ptr
L1
        copy    bx,c(ax)
;;            negative means end of data
        cmp     bx,#0
        jb      L2
        add     bx,sum
        copy    sum,bx
;;            each data is 2 byte long
        add     ax,#2
        jmp     L1
L2      copy    bx,sum
        out     bx
        end
sum     0
ptr     data
data    1
        2
        3
        4
        5
        -1
```

A03.p88

```
    IN AX
    CALL P1
    OUT  AX
    END
P1  ADD  AX,#10
    CALL P2
    SUB  AX,#10
    RTN
P2  COPY BX,AX
    ADD  BX,#1
    MUL  AX,BX
    RTN
```

A04.p88

```
;; factorial
        in      ax
        call    fact
        out     ax
        end
fact    cmp     ax,#2
        jb      L1
        push    ax
        sub     ax,#1
        call    fact
        pop     bx
        mul     ax,bx
        rtn
L1      copy    ax,#1
        rtn
```

```
;; Chapter 9 sample program in the text
        IN      AX
        OUT     AX
```

sample01.p88

```
;; Chapter 9 sample program in the text
        in      ax
        copy    m1,ax
        mul     ax,m1
        out     ax
```

sample02.p88

```
;; Chapter 9 sample program in the text
        in      ax
        copy    a,ax
        in      ax
        copy    b,ax
        copy    ax,a
        div     ax,b
        out     ax
```

sample03.p88

```
;; Chapter 9 sample program in the text
;; You should interrupt the infinite loop.
L1      ADD     AX,A
        JMP     L1
```

sample04.p88

```
;; Chapter 9 sample program
;; Data M0, M1, M10 are added.
        COPY    AX,M0
L1      ADD     AX,M1
        OUT     AX
        CMP     AX,M10
        JB      L1
        END
M0      0
M1      1
M10     10
```

sample05.p88

```
;; Chapter 9 sample program
;; Data are coded in immediate data
        COPY    AX,#0
L1      ADD     AX,#1
        OUT     AX
        CMP     AX,#10
        JB      L1
```

sample05b.p88

```
;; Chapter 9 sample program in the text
        in      ax
        copy    m1,ax
        sub     ax,m1
        cmp     ax,m1
;;          if ax < m1 then goto next
        jb      next
        sub     ax,m1
        copy    m1,ax
next    copy    ax,m1
        out     ax
```

sample06.p88

```
;; Chapter 9 sample program in the text
        IN      AX
        COPY    M1,AX
        SUB     AX, M1
        COPY    ZERO,AX
        COPY    SUM,AX
        COPY    AX,M1
LOOP    CMP     AX, ZERO
        JB      FIN
        ADD     AX,SUM
        COPY    SUM,AX
        IN      AX
        JMP     LOOP
FIN     COPY    AX, SUM
        OUT     AX
```

sample07.p88

```
;;      hanoi(a,b,c,n)
;;              hanoi(a,c,b, n-1)
;;              write("a->b")
;;              hanoi(c,b,a n-1)
        in      ax
        push    ax          ; 4th arg
        copy    ax,#2
        push    ax          ; 3rd arg
        copy    ax,#3
        push    ax          ; 2nd arg
        copy    ax,#1
        push    ax          ; 1st arg
        call    hanoi
        add     dx,#8 ; pop x4
        end

hanoi   push    cx
        copy    cx,dx ; stack frame
        copy    ax,10+c(cx) ; 4th arg
        sub     ax,#1
        cmp     ax,#1
        jb      wrt

        push    ax          ; new 4th arg
        copy    ax,6+c(cx)  ; 2nd arg
        push    ax          ; new 3rd arg
        copy    ax,8+c(cx)  ; 3rd arg
        push    ax          ; new 2nd arg
        copy    ax,4+c(cx)  ; 1st arg
        push    ax          ; new 1st arg
        call    hanoi
        add     dx,#8 ; pop x4

wrt     copy    ax,4+c(cx)  ; 1st arg
        mul     ax,#100
        copy    bx,6+c(cx)  ; 2nd arg
        add     ax,bx
        out     ax

        copy    ax,10+c(cx) ; 4th arg
        sub     ax,#1
        cmp     ax,#1
        jb      endhanoi

        push    ax          ; new 4th arg
        copy    ax,4+c(cx)  ; 1st arg
        push    ax          ; new 3rd arg
        copy    ax,6+c(cx)  ; 2nd arg
        push    ax          ; new 2nd arg
        copy    ax,8+c(cx)  ; 3rd arg
        push    ax          ; new 1st arg
        call    hanoi
        add     dx,#8 ; pop x4

endhanoi
        pop     cx
        rtn
```

```
; Linear List
LOOP
        in      ax
        cmp     ax,#0
        jb      WRT
        copy    data,ax
        copy    ax,root
SRCH
        copy    bx,2+c(ax); bx = ax->next
        cmp     bx,#0     ; if bx == null
        jb      INSERT
        copy    cx,c(bx)  ; cx = bx->data;
        cmp     cx,data
        jnb     INSERT    ; if (bx->data > data)
        copy    ax,bx     ; ax = bx
        jmp     SRCH
INSERT  copy    cx,ptr
        copy    dx,data
        copy    c(cx),dx
        copy    2+c(cx),bx
        copy    2+c(ax),cx
        add     cx,#4
        copy    ptr,cx
        jmp     LOOP

WRT
        copy    ax,root
WLOOP   copy    ax,2+c(ax)
        cmp     ax,#0
        jb      EXIT
        copy    bx,c(ax)
        out     bx
        jmp     WLOOP
EXIT    END

data    0
root    dummy
dummy   0
        -1              ; null
ptr     pool
pool    res     400     ; max 100 items
```