5. CALL, 条件JUMP命令を学ぼう

（プログラムカウンタとスタックの働き）

本稿のWebページ

http://www.mybook-pub-site.sakura.ne.jp/PIC/index.html

このソースファイルを打ち込んで下さい.

```
; Decrement f, Skip if 0

        INCLUDE"p16F84A.inc"
        list p=16F84A

        __CONFIG  _HS_OSC  &  _WDT_OFF  &  _PWRTE_OFF  &  _CP_OFF

MEM1    EQU     0x0C            ;MEM1 at 0C

        ORG     0
        GOTO    START
        ORG     4

START

        MOVLW   0x0A            ; Load 0x0A to W

        MOVWF   MEM1            ; Move W to MEM1

REPT    DECFSZ  MEM1,1          ; MEM1 - 1 -> MEM1, Skip next line if 0

        GOTO    REPT            ; Go to REPT

        GOTO    START

        END
```

このプログラムはMEM1に0x0Aの値を転送し，その数を一つずつ減らして行き，ゼロになったらSTARTにもどることを実行します.

```
; Decrement f, Skip if 0

            INCLUDE"p16F84A.inc"
            list p=16F84A

            __CONFIG   _HS_OSC  &   _WDT_OFF  &   _PWRTE_OFF  &   _CP_OFF

MEM1        EQU        0x0C                  ;MEM1 at 0C

            ORG        0
            GOTO       START
            ORG        4

START

            MOVLW   0x0A

            MOVWF   MEM1

REPT        DECFSZ   MEM1,1

            GOTO       REPT

            GOTO       START

            END
```

MEM1の内容を1減らして，その結果をMEM1に入れます．もし，結果が0であれば次の命令を飛ばして，その次の命令を実行します．結果が0でない限り，次の命令を実行します．

DECFSZ　f, d　　Decrement f, skip if zeroの略

f – 1の結果を d に転送します．そして，結果が 0 であれば次の命令を飛ばして，その次の命令を実行します．

```
; Decrement f, Skip if 0

            INCLUDE"p16F84A.inc"
            list p=16F84A

            __CONFIG   _HS_OSC  &   _

MEM1        EQU        0x0C

            ORG        0
            GOTO       START
            ORG        4

START

            MOVLW   0x0A

            MOVWF   MEM1

REPT        DECFSZ   MEM1,1

            GOTO       REPT

            GOTO       START

            END
```

MEM1の内容が 0 になるまで同じ演算をくり返します．これは例えば次の例のように

の中の処理を複数回繰り返して実行させる場合などに便利です．

このソースファイルを打ち込んで下さい.

```
; Call

        INCLUDE"p16F84A.inc"
        list p=16F84A

        __CONFIG  _HS_OSC  &  _WDT_OFF  &  _PWRTE_OFF  &  _CP_OFF

MEM1    EQU     0x0C            ;MEM1 at 0C

        ORG     0
        GOTO    START
        ORG     4

START
        MOVLW   0x0A            ; Load 0x0A to W

        MOVWF   MEM1            ; Move W to MEM1

        CALL    REPT            ; Subroutine CALL

        GOTO    START


REPT    DECFSZ  MEM1,1          ; MEM1 - 1 -> MEM1, Skip next if 0

        GOTO    REPT            ; Go to REPT

        RETURN

        END
```

このプログラムはサブルーチンコールを行います.

```
; Call

            INCLUDE"p16F84A.inc"
            list p=16F84A

            __CONFIG  _HS_OSC  &  _WDT_OFF  &  _PWRTE_OFF  &  _CP_OFF

MEM1        EQU       0x0C                    ;MEM1 at 0C

            ORG       0
            GOTO      START
            ORG       4

START
            MOVLW   0x0A

            MOVWF     MEM1

            CALL      REPT

            GOTO      START

REPT        DECFSZ    MEM1,1

            GOTO      REPT

            RETURN

            END
```

メインルーチン

サブルーチン

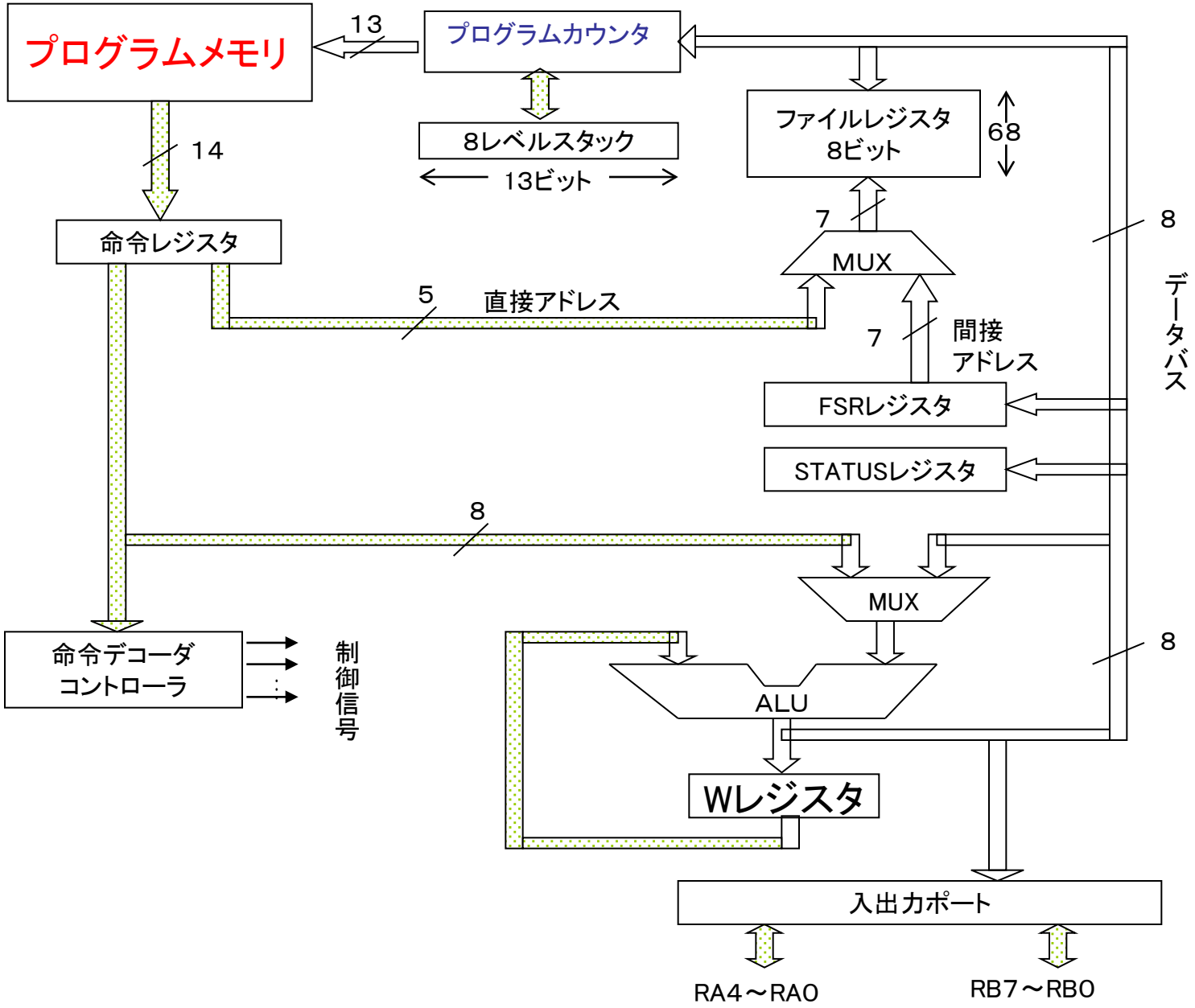
```
; Call

            INCLUDE"p16F84A.inc"
            list p=16F84A

            __CONFIG  _HS_OSC  &  _WDT_OFF  &  _PWRTE_OFF  &  _CP_OFF

MEM1        EQU       0x0C

            ORG       0
            GOTO      START
            ORG       4

START
            MOVLW   0x0A

            MOVWF     MEM1

            CALL      REPT

            GOTO      START

REPT        DECFSZ    MEM1,1

            GOTO      REPT

            RETURN

            END
```

CALL文によりREPT番地へ跳びます．RETURN命令によりCALL文の一つ下の命令にもどってきます．

CALL　kk　Call subroutineの略

プログラム中のあちこちで同じ処理をさせる必要がある場合に，一つだけサブルーチンを書き，このサブプルーチンをあちこちでコールするだけで良いので便利です．

PIC16F84AのData SheetのBLOCK DIAGRAMの抜粋です.



さて，ここでプログラムメモリ内に格納されている内容を見てみましょう．
以下は，本章p.5のプログラムを打ち込み，そのプロジェクト名をcall.mcp，ファイル名をcall.asmとして，Project → MakeによりBULD SUCCEEDEDのメッセージを得た段階において，call.mcp, call.asmと同じフォルダ内に生成されたcall.lstファイルの抜粋です．

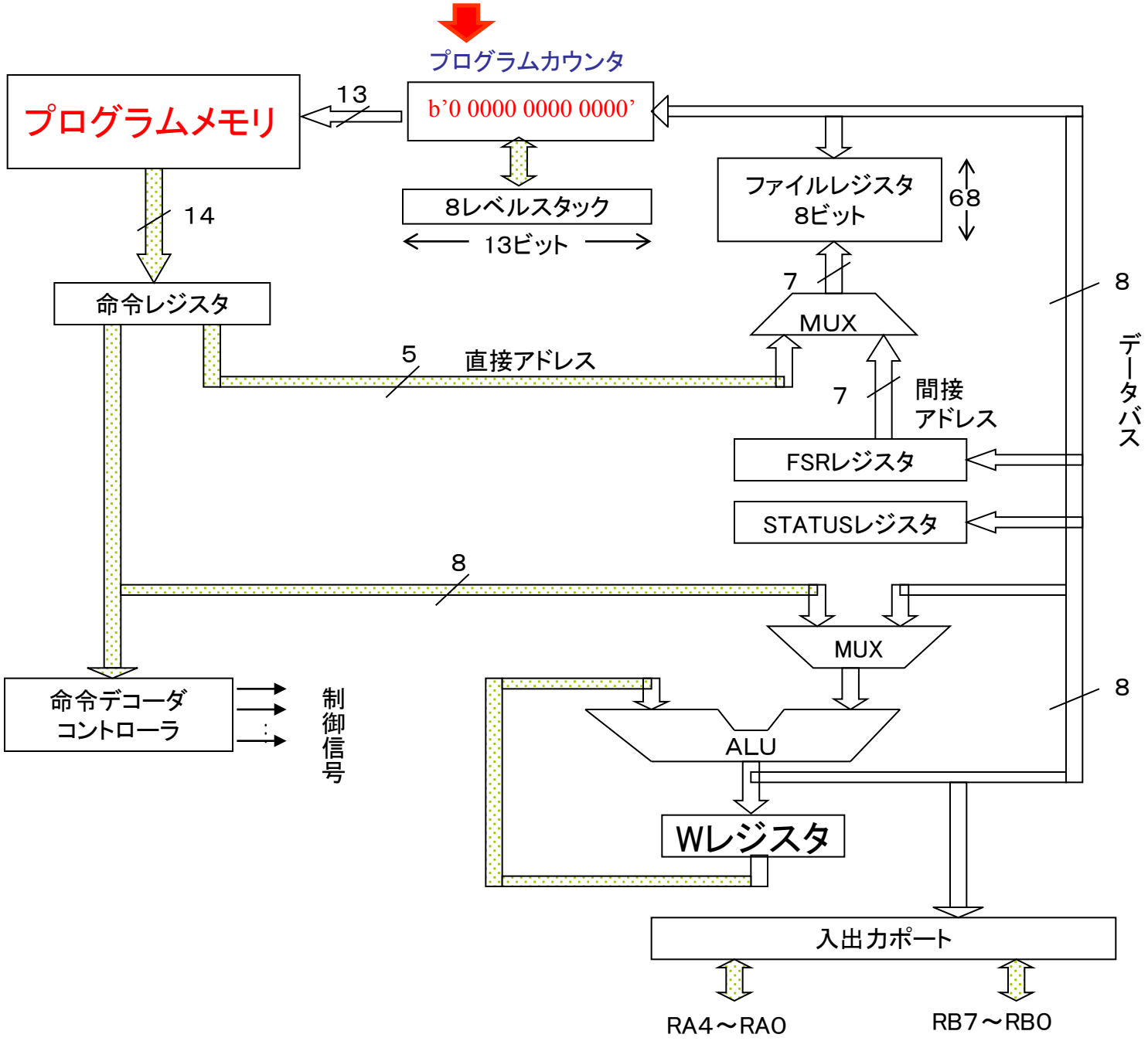
```
LOC  OBJECT CODE     LINE SOURCE TEXT


0000                 00010             ORG     0
0000  2804            00011             GOTO    START
0004                 00012             ORG     4
                     00013
0004                  00014 START
0004  300A           00015             MOVLW   0x0A          ; Load 0x0A to W
                     00016
0005  008C           00017             MOVWF   MEM1          ; Move W to MEM1
                     00018
0006  2008           00019             CALL    REPT           ; Subroutine CALL
                     00020
0007  2804           00021             GOTO    START
                     00022
                     00023
0008  0B8C           00024 REPT    DECFSZ  MEM1,1          ; MEM1 - 1 -> MEM1, Skip next if 0
                     00025
0009  2808           00026             GOTO    REPT           ; Go to REPT
                     00027
000A  0008           00028             RETURN
                     00029
                     00030             END
```

左はオブジェクトコード(OBJECT CODE)とそのコードが格納されるプログラムメモリの番地(LOC, Locationの略)です．右はソーステキストです．例えばGOTO　START は0x28という機械語の命令に翻訳され，ジャンプ先の0x04番地の数値とつなげられて0x2804という数値に変換されて0x00番地に格納されます．STARTには0x04番地が割り当てられ，この番地に0x300Aが格納されます．0x30は MOVLWに対応する機械語であり，リテラル0x0Aとつなげられています．次の0x05番地の0x00はMOVWFを意味し，0x8CはMEM1の番地です．これは MEM1　EQU　0x0Cと定義されていることによります．なお，0x0Cと0x8Cは同じ番地です．

LOC	OBJECT CODE
0000	
0000	2804
0004	
0004	
0004	300A
0005	008C
0006	2008
0007	2804
0008	0B8C
0009	2808
000A	0008

```
LINE SOURCE TEXT


00010            ORG    0
00011            GOTO   START
00012            ORG    4
00013
 00014 START
00015            MOVLW  0x0A          ; Load 0x0A to W
00016
00017            MOVWF  MEM1          ; Move W to MEM1
00018
00019            CALL   REPT          ; Subroutine CALL
00020
00021            GOTO   START
00022
00023
00024 REPT   DECFSZ  MEM1,1           ; MEM1 - 1 -> MEM1, Skip next if 0
00025
00026            GOTO   REPT          ; Go to REPT
00027
00028            RETURN
00029
00030            END
```

マイコンの電源がオンとなると，プログラムカウンタは0x0000にリセットされます．



プログラムカウンタが0x0000のとき，プログラムメモリの0x0000番地の0x2804が命令レジスタに読み出されます．0x2804は2進表現ではb’10 10000 0000 0100’です．上位6ビットが命令を表し，下位8ビットがこの場合は番地を表しています．上位6ビットが命令デコーダで解読され，これがGOTO命令であるので，下位8ビットの0x04がプログラムカウンタに送られます．プログラムカウンタの中身が0x0004となることで，次にはプログラムメモリの0x0004番地の0x300Aが命令レジスタに読み出されます．以降，プログラムカウンタは一つずつインクリメントされ，プログラムメモリ内の命令が順次実行されます．

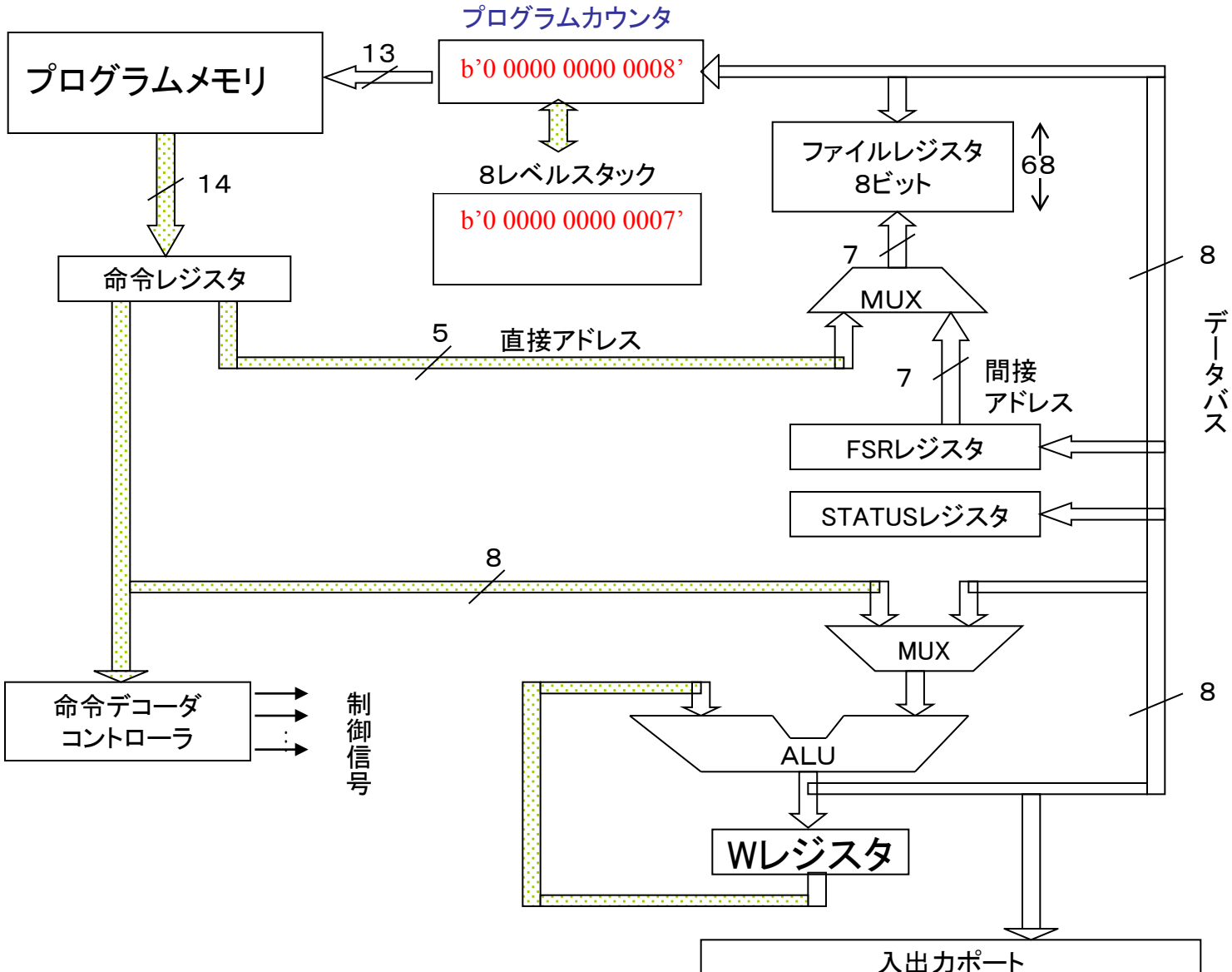
```
LOC  OBJECT CODE     LINE SOURCE TEXT


0000                 00010             ORG     0
0000 2804            00011             GOTO    START
0004                 00012             ORG     4
                     00013
0004                  00014 START
0004 300A            00015             MOVLW   0x0A          ; Load 0x0A to W
                     00016
0005 008C            00017             MOVWF   MEM1          ; Move W to MEM1
                     00018
0006 2008            00019             CALL    REPT           ; Subroutine CALL
                     00020
0007 2804            00021             GOTO    START
                     00022
                     00023
0008 0B8C            00024 REPT    DECFSZ  MEM1,1           ; MEM1 - 1 -> MEM1, Skip next if 0
                     00025
0009 2808            00026             GOTO    REPT           ; Go to REPT
                     00027
000A  0008           00028             RETURN
                     00029
                     00030             END
```

0x06番地の0x2008はCALL REPTに対応します．0x20がCALL命令の機械語です．REPTが0x08番地に割り当てられていますので，0x20の後に0x08がつけられています．

LOC	OBJECT CODE
0000	
0000	2804
0004	
0004	
0004	300A
0005	008C
0006	2008
0007	2804
0008	0B8C
0009	2808
000A	0008

```
LINE SOURCE TEXT

00010            ORG    0
00011            GOTO   START
00012            ORG    4
00013
 00014 START
00015            MOVLW  0x0A          ; Load 0x0A to W
00016
00017            MOVWF  MEM1          ; Move W to MEM1
00018
00019            CALL   REPT          ; Subroutine CALL
00020
00021            GOTO   START
00022
00023
00024 REPT   DECFSZ  MEM1,1           ; MEM1 - 1 -> MEM1, Skip next if 0
00025
00026            GOTO   REPT          ; Go to REPT
00027
00028            RETURN
00029
00030            END
```

0x06番地の0x2008が実行されると，プログラムカウンタの値は0x0008とされ，また，CALL命令の一つ下の番地0x07がスタックに保存されます．次には0x08番地の0x0B8Cが実行されます．その後，RETURN命令が実行されると，このスタック内の番地0x07がプログラムカウンタに転送され，次には0x07番地の命令が実行されます．



演習問題12. MEM1に５をロードし，MEM2に３をロードし，MEM1とMEM2の内容のかけ算の結果をMEM3に格納するプログラムを作成せよ.

ヒント１：５×３のかけ算は５＋５＋５を実行すればよい.

ヒント２：DECFSZ　MEM2,1　と　ADDWF MEM1,0

を組み合わせればよい.

演習問題13. 上の問題において「MEM1とMEM2の内容のかけ算を実行する」部分をサブルーチンとし，５×３の結果をMEM3に　２×5の結果をMEM4に格納するプログラムを作成せよ.

演習問題12　解答例

```
; Multiplication of MEM1 and MEM2

            INCLUDE"p16F84A.inc"
            list p=16F84A

            __CONFIG  _HS_OSC  &  _WDT_OFF  &  _PWRTE_OFF  &  _CP_OFF

MEM1        EQU       0x0C                ;MEM1 at 0C
MEM2        EQU       0x0C+1              ;MEM2 at 0D
MEM3        EQU       0x0C+2              ;MEM3 at 0E

            ORG       0
            GOTO      START
            ORG       4

START       MOVLW     0x05                ; Load 0x05 to W
            MOVWF     MEM1                ; Move W to MEM1

            MOVLW     0x03                ; Load 0x03 to W
            MOVWF     MEM2                ; Move W to MEM2

            MOVLW     0x00
REPT        ADDWF     MEM1,0              ; W + MEM1 -> W
            DECFSZ    MEM2,1              ; MEM1 - 1 -> MEM1, Skip next line if 0
            GOTO      REPT

            MOVWF     MEM3                ; Move W to MEM3
            GOTO      START

            END
```

演習問題１３　解答例

```
; Call a subroutine of Multiplication of MEM1 and MEM2

            INCLUDE"p16F84A.inc"
            list p=16F84A

            __CONFIG  _HS_OSC  &  _WDT_OFF  &  _PWRTE_OFF  &  _CP_OFF

MEM1        EQU       0x0C                ;MEM1 at 0C
MEM2        EQU       0x0C+1              ;MEM2 at 0D
MEM3        EQU       0x0C+2              ;MEM3 at 0E
MEM4        EQU       0x0C+3              ;MEM2 at 0F

            ORG       0
            GOTO      START
            ORG       4
```

（次ページに続く）

```
START

;Main Routine

            MOVLW   0x05            ; Load 0x05 to W
            MOVWF   MEM1            ; Move W to MEM1
            MOVLW   0x03            ; Load 0x03 to W
            MOVWF   MEM2            ; Move W to MEM2
            CALL    MUL
            MOVWF   MEM3

            MOVLW   0x02            ; Load 0x05 to W
            MOVWF   MEM1            ; Move W to MEM1
            MOVLW   0x05            ; Load 0x03 to W
            MOVWF   MEM2            ; Move W to MEM2
            CALL    MUL
            MOVWF   MEM4

            GOTO    START

;Sub Routine

MUL         MOVLW    0x00
REPT        ADDWF   MEM1,0          ; W + MEM1 -> W
            DECFSZ  MEM2,1          ; MEM1 - 1 -> MEM1, Skip next line if 0
            GOTO    REPT
            RETURN

            END
```

演習問題14. MEM1に８をロードし，MEM2に３をロードし，MEM1とMEM2の内容の割算の商をMEM3に，余りをMEM4に格納するプログラムを作成せよ.

ヒント１：８÷３の割算は８－３－３を実行すればよい.

ヒント２：８－３と２－３では桁上がりが違う.

演習問題15. 上の問題において「MEM1とMEM2の内容の割算を実行する」部分をサブルーチンとし，８÷３の結果をMEM3，MEM4に　100÷23の結果をMEM5,MEM6に格納するプログラムを作成せよ.

演習問題16. 以下は時間稼ぎサブルーチンである．このサブルーチンの使い道にはどのようなものが考えられるか．列挙せよ.

```
COUNT1  MOVLW   0x45
        MOVWF   TIME1
STEPM   DECFSZ  TIME1,1
        GOTO    STEPM
        RETURN
```

演習問題14　解答例

```
; Division of MEM1 by MEM2

            INCLUDE"p16F84A.inc"
            list p=16F84A

            __CONFIG  _HS_OSC  &  _WDT_OFF  &  _PWRTE_OFF  &  _CP_OFF

MEM1        EQU       0x0C                ;MEM1 at 0C
MEM2        EQU       0x0C+1              ;MEM2 at 0D
MEM3        EQU       0x0C+2              ;MEM3 at 0E
MEM4        EQU       0x0C+3              ;MEM4 at 0F
TEMP        EQU       0x0C+4              ;TEMP at 10

            ORG       0
            GOTO      START
            ORG       4
```

（次ページへ続く）

```
START

        MOVLW   0x08            ; Load 0x08 to W
        MOVWF   MEM1            ; Move W to MEM1

        MOVLW   0x03            ; Load 0x03 to W
        MOVWF   MEM2            ; Move W to MEM2



        MOVF    MEM1,0          ;MEM1 -> W
        MOVWF   TEMP            ;W -> TEMP
        MOVF    MEM2,0          ;MEM2 -> W
        CLRF    MEM3            ;0 -> MEM3
REPT    SUBWF   TEMP,1          ; TEMP - W -> TEMP
        BTFSS   STATUS,0        ;Skip next line if carry = 1
        GOTO    DIVEND
        SUBWF   MEM1,1          ;MEM1 - W -> MEM1
        INCF    MEM3,1          ;MEM3 + 1 -> MEM3
        GOTO    REPT

DIVEND  MOVF    MEM1,0          ; MEM1 -> W
        MOVWF   MEM4            ; W -> MEM4
        GOTO    START

        END
```

TEMPレジスタの内容がマイナスとなるまで
(MEM1) – (MEM2)
の計算を繰り返す.
引いた回数はMEM3をインクリメントすることで数えておく. これがそのまま商となる.
MEM1に残ったものは余りである.

演習問題15　解答例

```
; Call a Subroutine of Division of MEM1 by MEM2

            INCLUDE"p16F84A.inc"
            list p=16F84A

            __CONFIG  _HS_OSC  &  _WDT_OFF  &  _PWRTE_OFF  &  _CP_OFF

MEM1        EQU       0x0C              ;MEM1 at 0C
MEM2        EQU       0x0C+1            ;MEM2 at 0D
MEM3        EQU       0x0C+2            ;MEM3 at 0E
MEM4        EQU       0x0C+3            ;MEM4 at 0F
MEM5        EQU       0x0C+4            ;MEM5 at 10
MEM6        EQU       0x0C+5            ;MEM6 at 11
TEMP1       EQU       0x0C+6            ;TEMP1 at 12
TEMP2       EQU       0x0C+7            ;TEMP2 at 13


            ORG       0
            GOTO      START
            ORG       4
```

（次ページへ続く）

```
;Main Routine

START   MOVLW   0x08            ; Load 0x08 to W
        MOVWF   MEM1            ; Move W to MEM1
        MOVLW   0x03            ; Load 0x03 to W
        MOVWF   MEM2            ; Move W to MEM2

        CALL    DIV

        MOVF    TEMP2,0         ;TEMP2 -> W
        MOVWF   MEM3            ;W -> MEM3
        MOVF    MEM1,0          ;MEM1 -> W
        MOVWF   MEM4            ;W -> MEM4


        MOVLW   D'100'          ; Load D'100' to W
        MOVWF   MEM1            ; Move W to MEM1
        MOVLW    D'23'          ; Load D'23' to W
        MOVWF   MEM2            ; Move W to MEM2

        CALL    DIV

        MOVF    TEMP2,0         ;TEMP2 -> W
        MOVWF   MEM5            ;W -> MEM5
        MOVF    MEM1,0          ;MEM1 -> W
        MOVWF   MEM6            ;W -> MEM6

        GOTO    START
```

（次ページへ続く）

```
;Subroutine

DIV     MOVF    MEM1,0          ;MEM1 -> W
        MOVWF   TEMP1           ;W -> TEMP1
        MOVF    MEM2,0          ;MEM2 -> W
        CLRF    TEMP2           ;0 -> TEMP2
REPT    SUBWF   TEMP1,1         ; TEMP1 - W -> TEMP1
        BTFSS   STATUS,0        ;Skip next line if carry = 1
        GOTO    RET
        SUBWF   MEM1,1          ;MEM1 - W -> MEM1
        INCF    TEMP2,1         ;TEMP2 + 1 -> TEMP2
        GOTO    REPT

RET     RETURN

        END
```

演習問題16　解答例

```
; Call a Subroutine for Time Consuming

            INCLUDE"p16F84A.inc"
            list p=16F84A

            __CONFIG  _HS_OSC  &  _WDT_OFF  &  _PWRTE_OFF  &  _CP_OFF

TIME1       EQU       0x0C                    ;TIME1 at 0C

            ORG       0
            GOTO      START
            ORG       4

START

            CALL      COUNT1

            GOTO      START

COUNT1      MOVLW     0x45
            MOVWF     TIME1
STEPM       DECFSZ    TIME1,1
            GOTO      STEPM
            RETURN

            END
```

2004年8月

著者： 古橋武
名古屋大学工学研究科計算理工学専攻
furuhashi@cse.nagoya-u.ac.jp