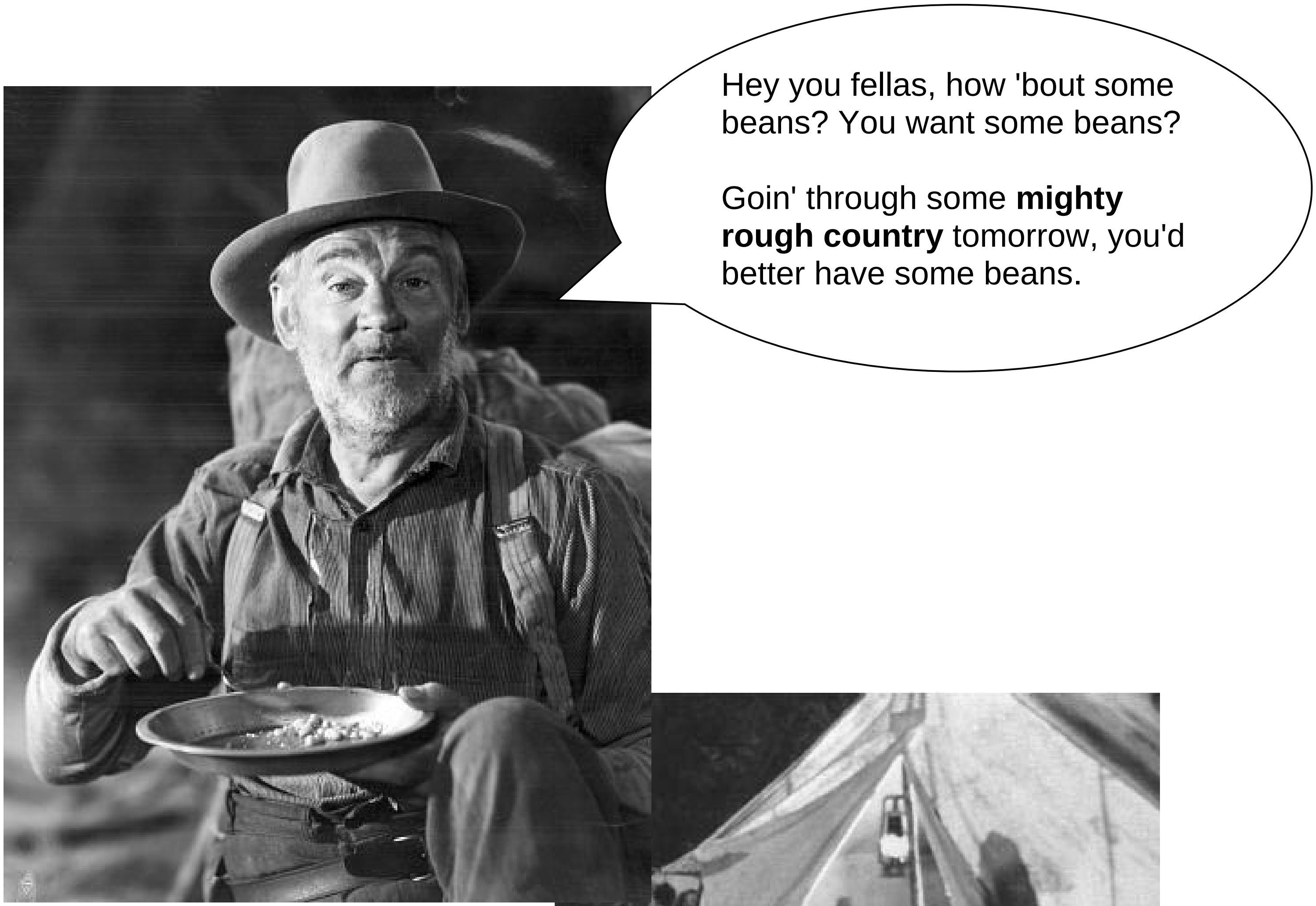
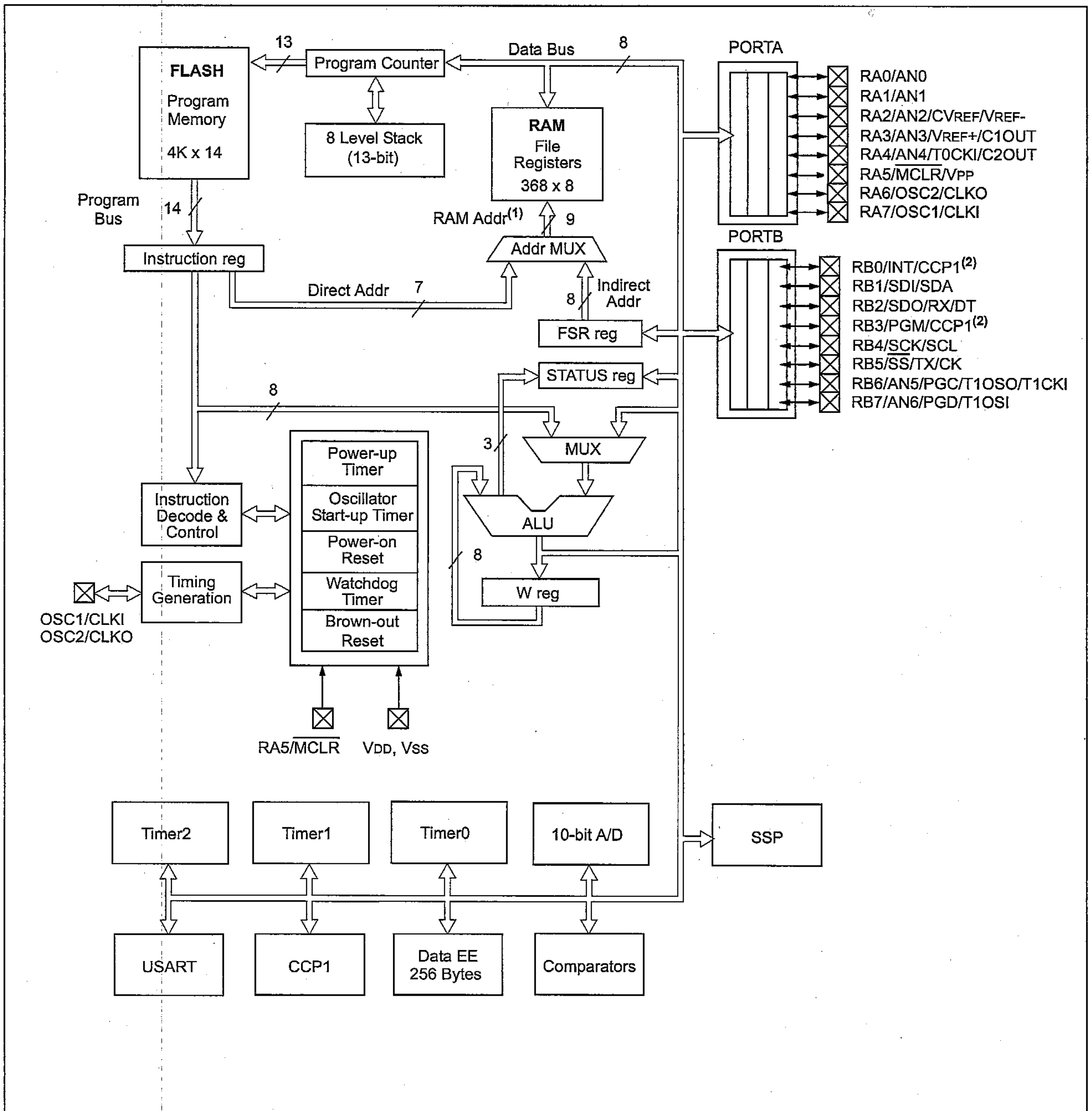

Hey you fellas, how 'bout some beans? You want some beans?
Goin' through some **mighty rough country** tomorrow, you'd better have some beans.

FIGURE 1-2: PIC16F88 DEVICE BLOCK DIAGRAM



Note 1: Higher order bits are from the STATUS register.

2: The CCP1 pin is determined by CCPMX in Configuration Word 1 register.

MICRO <u>PROCESSORS</u> / MICRO <u>CONTROLLERS</u>

<u>HUGE</u> TOPIC - WE'LL SCRATCH SURFACE

IDEED, ONLY PARTLY LEARN OUR <u>OWN</u> DEVICE

FOCUS ON

- ARCHITECTURE - WHAT HAPPENS INSIDE
- I/O - CONNECTING TO EXTERAL CIRCUITS
- PROGRAMMING - <u>ASSEMBLY</u> LANGUAGE

JARGON:

"MICRO" — ALL ON ONE CHIP

"PROCESSOR" — ALU + LOCAL MEMORY + REGISTERS

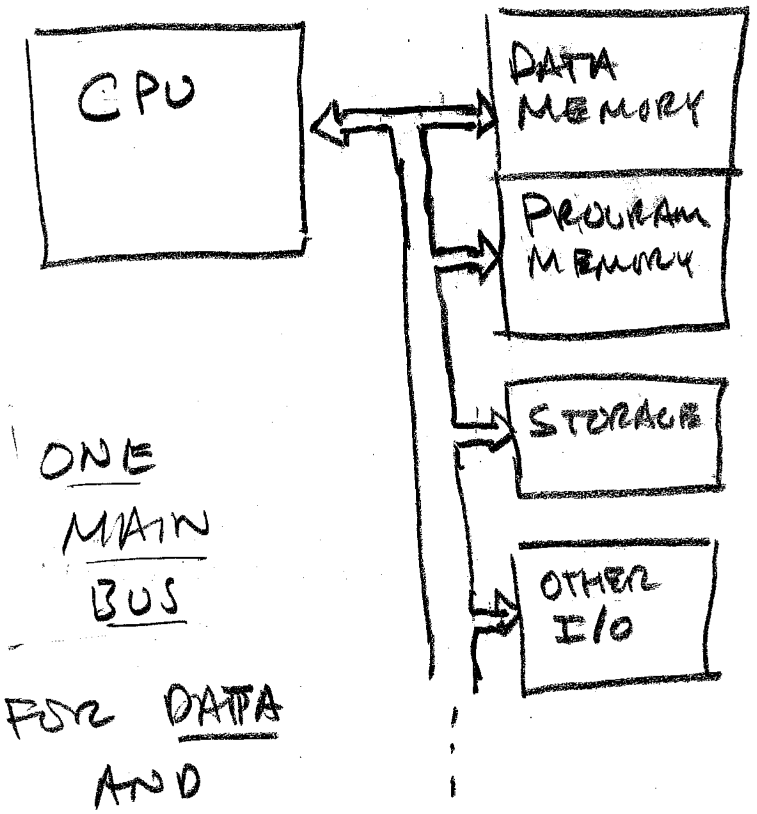
"CONTROLLER" — LIMITED MICROPROCESSOR, DESIGNED FOR SPECIFIC TASKS

"ALU" = ARITHMETIC LOGIC UNIT, DOES ADD, SUBTRACT, AND, OR, ... ON REGISTERS

MICROPROCESSOR	MICRO CONTROLLER
General purpose - DESKTOP / LAPTOP / DATA CENTER	"EMBEDDED" — OFTEN INVISIBLE APPLIANCES, CARS, CARD READERS, INDUSTRIAL CONTROLS
LARGE DATA & MEMORY CAPACITY <u>32 bit</u>, <u>64 bit</u>	LIMITED MEMORY / DATA <u>8 bit - 32 bit</u>
SPECIAL FUNCTIONS <u>NOT</u> <u>ONBOARD</u>	SPECIAL FUNCTIONS ONBOARD: E.G. SERIAL COMM, TIMERS, A/D, etc

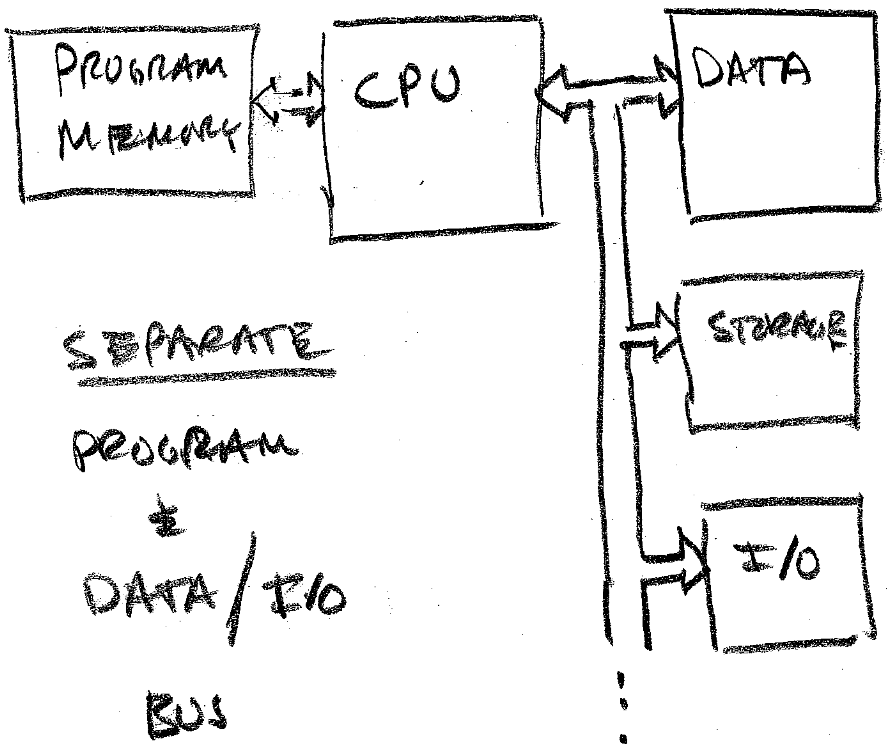
OVERALL ARCHITECTURE

"VON NEUMANN"



ONE MAIN BUS FOR DATA AND ADDRESS

"HARVARD"

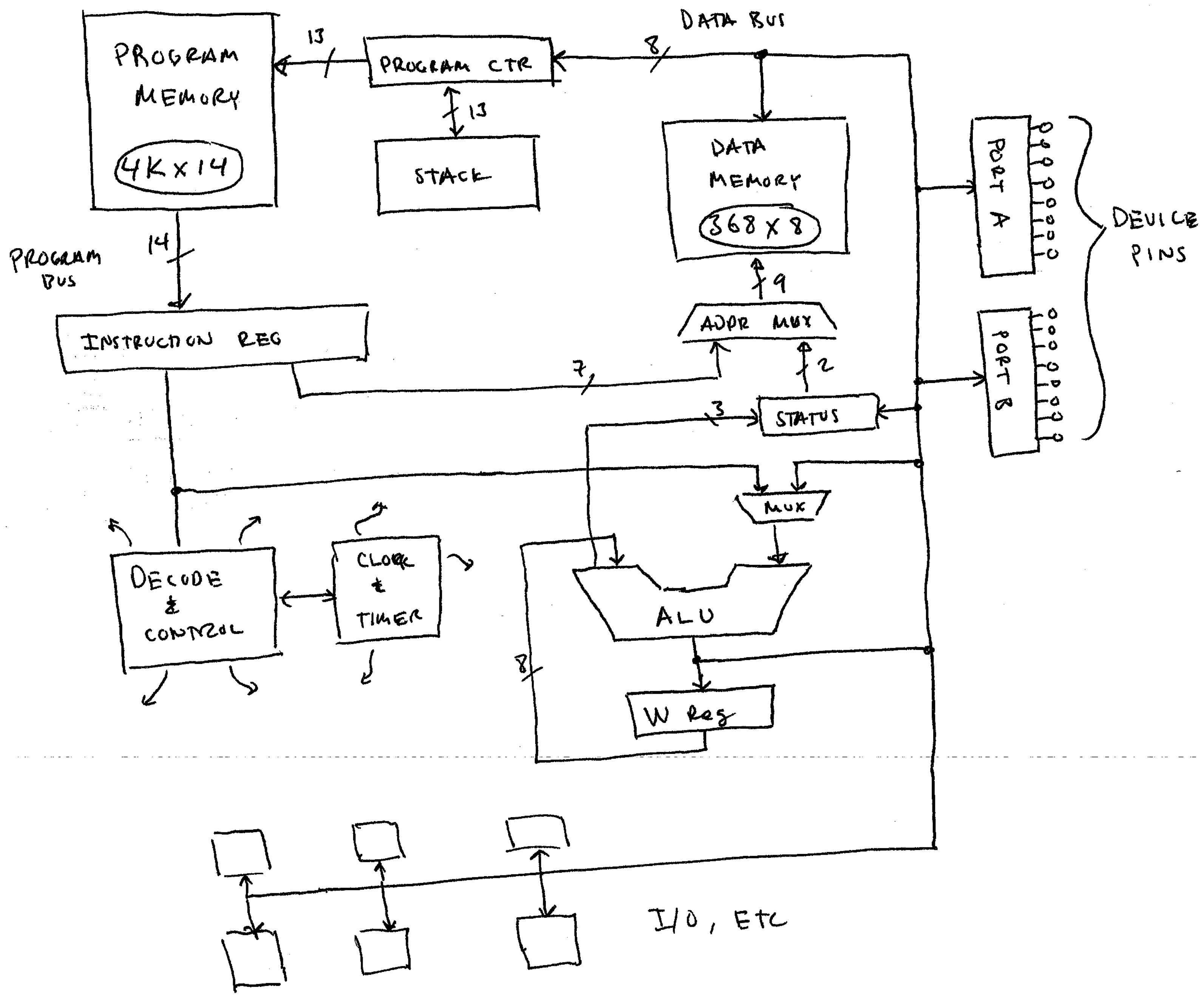


SEPARATE PROGRAM & DATA/I/O BUS

ADVANTAGE: SIMPLER DESIGN ESPECIALLY FOR LARGE PROGRAMS & ADAPTABLE CODING

ADVANTAGE: INHERENTLY FASTER – CAN RUN BOTH DATA & PROGRAM OPS AT SAME TIME

SIMPLIFIED ARCHITECTURE OF PIC

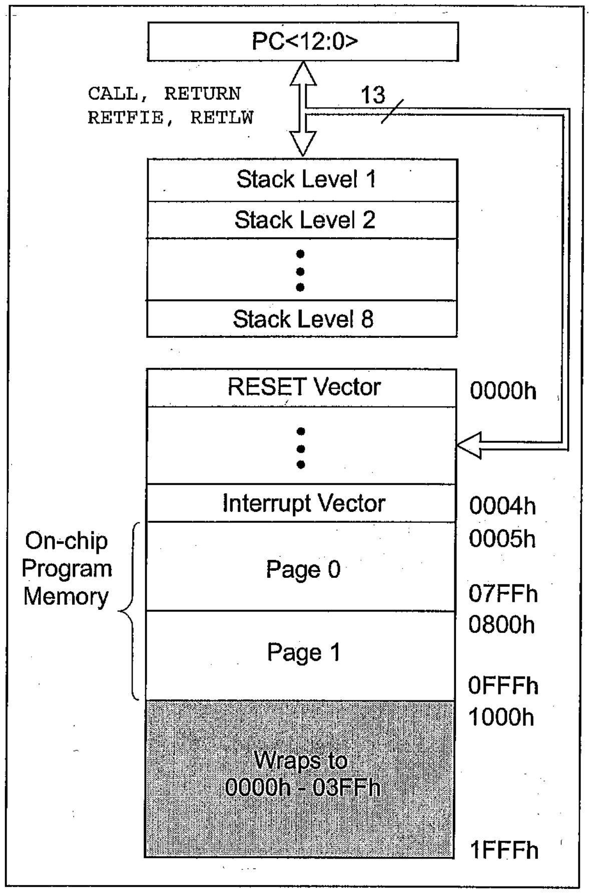


PROGRAM MEMORY

- ACCESSED SEQUENTIALLY*
- "FLASH" – NON-VOLATILE
- $\left\{ \begin{array}{l} \underline{4K} \times \underline{14} \\ \text{(14 bit width} \Rightarrow \text{SHORTER CYCLES)} \end{array} \right.$
- RESET → LOCATE 0000h
- INTERRUPT → EXECUTE AT 0004h
- CALLS / BRANCHES PUT PC VALUES ON "<u>STACK</u>"
- NOTE: PROGRAM <u>ADDRESS</u> IS 13 bits → 1FFFh "WRAPAROUND" above 1000h

* ALMOST → BRANCHES, INTERRUPTS CAUSE CHANGES

PROGRAM MEMORY MAP AND STACK: PIC16F87/88



DATA MEMORY

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Bits 5,6 in Status

- ACCESSED RANDOMLY
- VOLATILE (BUT CAN SAVE TO EEPROM)
- DIVIDED INTO 4 BANKS OF 128 bytes each (7Fh)
  ↳ BANKS ARE SELECTED BY STATUS REGISTER
- MUCH OF DATA MEMORY (UP TO 20h) IS TAKEN UP WITH SPECIAL FUNCTION REGISTERS
- Regular MEMORY = GENERAL PURPOSE REGISTERS

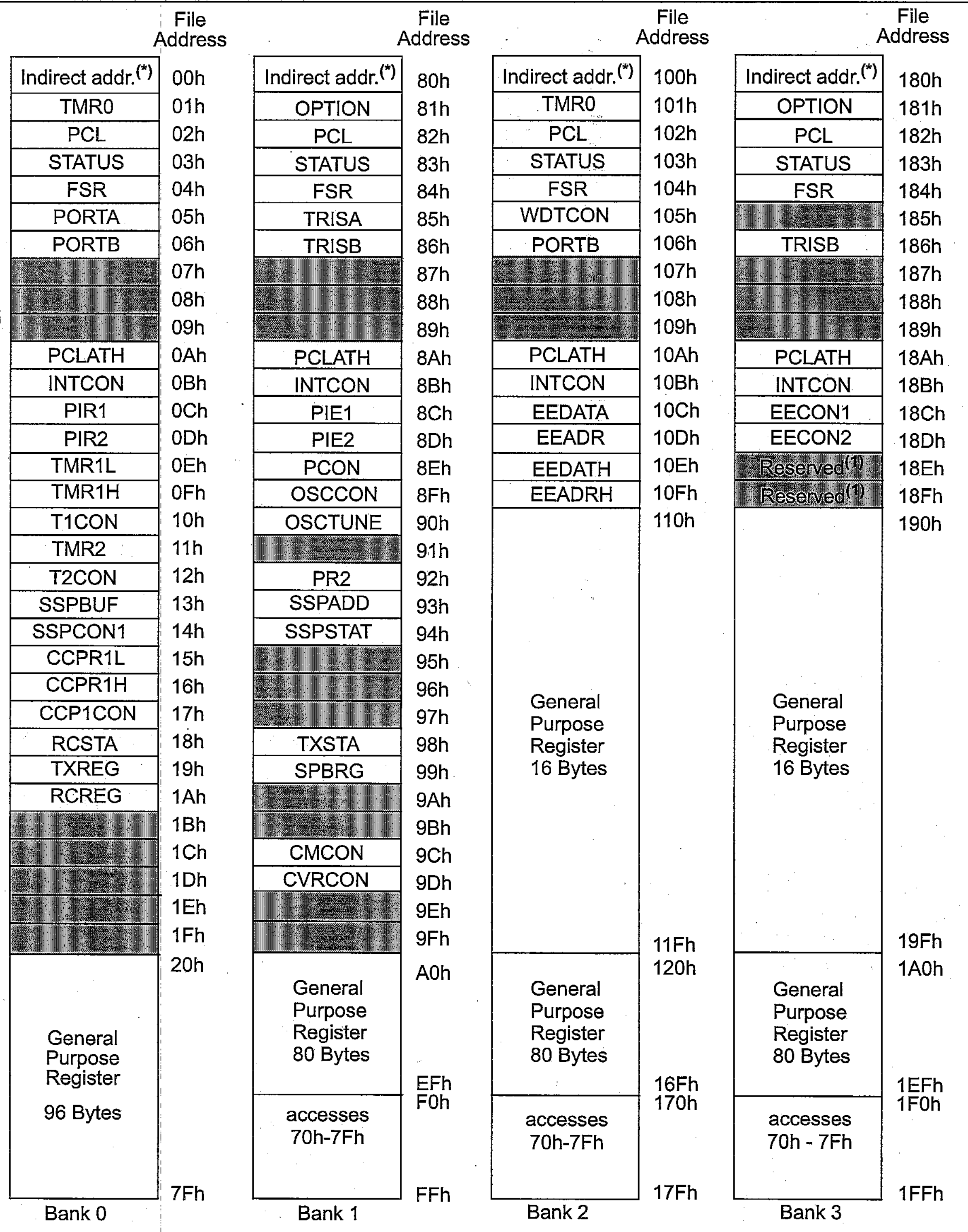
$\boxed{\text{!! SFR} \neq \text{GPR !!}}$

96 x 4 = 384 bytes data memory
(+ 256 bytes EEPROM)

2.2.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly, or indirectly through the File Select Register (FSR).

FIGURE 2-2: PIC16F87 REGISTER FILE MAP



	File Address		File Address		File Address		File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION	81h	TMR0	101h	OPTION	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h	WDTCON	105h	▨	185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
▨	07h	▨	87h	▨	107h	▨	187h
▨	08h	▨	88h	▨	108h	▨	188h
▨	09h	▨	89h	▨	109h	▨	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(1)	18Eh
TMR1H	0Fh	OSCCON	8Fh	EEADRH	10Fh	Reserved(1)	18Fh
T1CON	10h	OSCTUNE	90h	General Purpose Register 16 Bytes	110h	General Purpose Register 16 Bytes	190h
TMR2	11h	▨	91h				
T2CON	12h	PR2	92h				
SSPBUF	13h	SSPADD	93h				
SSPCON1	14h	SSPSTAT	94h				
CCPR1L	15h	▨	95h				
CCPR1H	16h	▨	96h				
CCP1CON	17h	▨	97h				
RCSTA	18h	TXSTA	98h				
TXREG	19h	SPBRG	99h				
RCREG	1Ah	▨	9Ah				
▨	1Bh	▨	9Bh				
▨	1Ch	CMCON	9Ch				
▨	1Dh	CVRCON	9Dh				
▨	1Eh	▨	9Eh				
▨	1Fh	▨	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
	7Fh		FFh		17Fh		1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

▨ Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: This register is reserved, maintain this register clear.

A VERY SPECIAL FUNCTION REGISTER: STATUS

LOCATION:

03h, 83h, 103h, 183h

USED BY

- ADDRESS DECODING
- POWER, CONTROL
- ALU OPS

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7	**IRP:** Register Bank Select bit (used for indirect addressing) 1 = Bank 2, 3 (100h - 1FFh) 0 = Bank 0, 1 (00h - FFh) For devices with only Bank0 and Bank1 the IRP bit is reserved, always maintain this bit clear.
bit 6:5	**RP1:RP0:** Register Bank Select bits (used for direct addressing) 11 = Bank 3 (180h - 1FFh) 10 = Bank 2 (100h - 17Fh) 01 = Bank 1 (80h - FFh) 00 = Bank 0 (00h - 7Fh) Each bank is 128 bytes. For devices with only Bank0 and Bank1 the IRP bit is reserved, always maintain this bit clear.
bit 4	**$\overline{TO}$:** Time-out bit 1 = After power-up, CLRWDT instruction, or SLEEP instruction 0 = A WDT time-out occurred
bit 3	**$\overline{PD}$:** Power-down bit 1 = After power-up or by the CLRWDT instruction 0 = By execution of the SLEEP instruction
bit2	**Z:** Zero bit 1 = The result of an arithmetic or logic operation is zero 0 = The result of an arithmetic or logic operation is not zero
bit 1	**DC:** Digit carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$ the polarity is reversed) 1 = A carry-out from the 4th low order bit of the result occurred 0 = No carry-out from the 4th low order bit of the result
bit 0	**C:** Carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) 1 = A carry-out from the most significant bit of the result occurred 0 = No carry-out from the most significant bit of the result occurred

Note: For $\overline{\text{borrow}}$ the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend	
R = Readable bit	W = Writable bit
U = Unimplemented bit, read as '0'	- n = Value at POR reset

ALU & W REGISTER

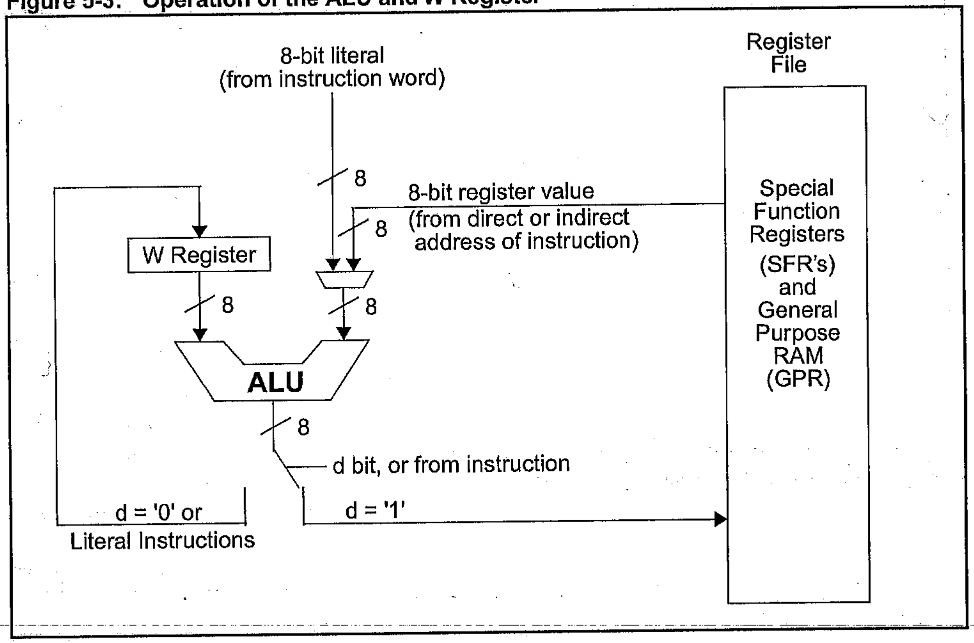
W or "WORKING" REGISTER IS LIKE A "SCRATCH-PAD"
CONTAINS TEMPORARY RESULTS OF ALU OPS

W SOMETIMES CALLED
"ACCUMULATOR"

ALU IS HEART OF
CPU - WHERE
CALCULATIONS MADE

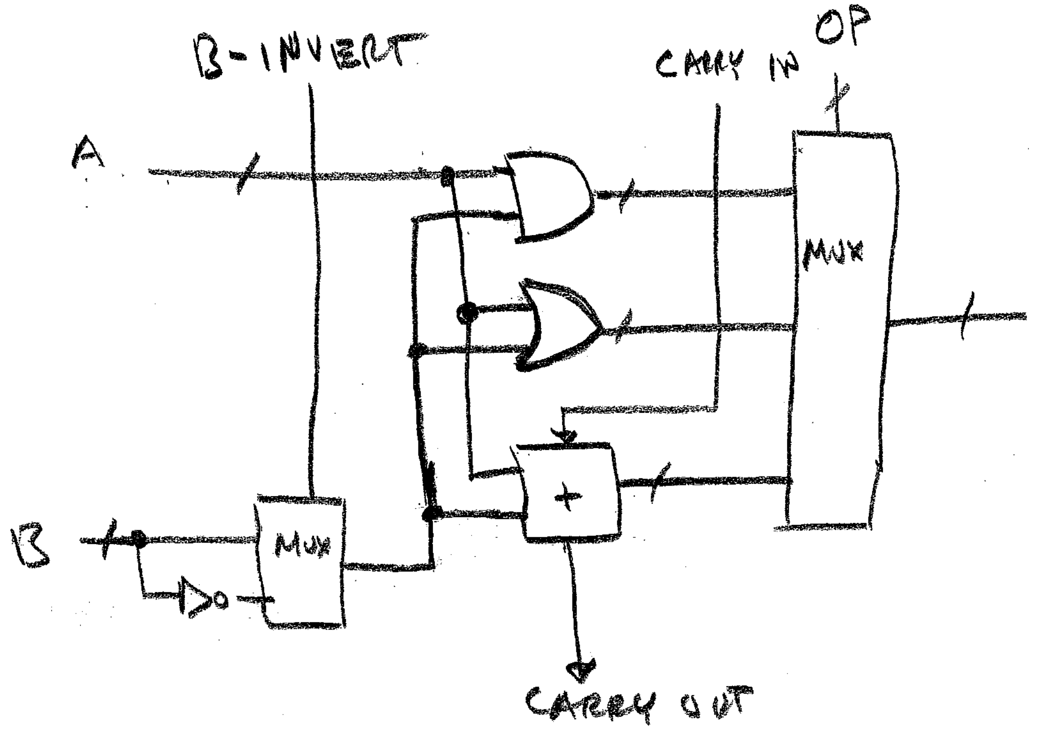
MOST OPS HAVE
DESTINATION OF W
OR MEMORY F

Figure 5-3: Operation of the ALU and W Register



8-bit literal
(from instruction word)
Register File
8
8-bit register value
(from direct or indirect address of instruction)
8
W Register
8
8
ALU
8
d bit, or from instruction
d = '0' or
Literal Instructions
d = '1'
Special Function Registers (SFR's) and General Purpose RAM (GPR)

A PEEK INSIDE AN ALU



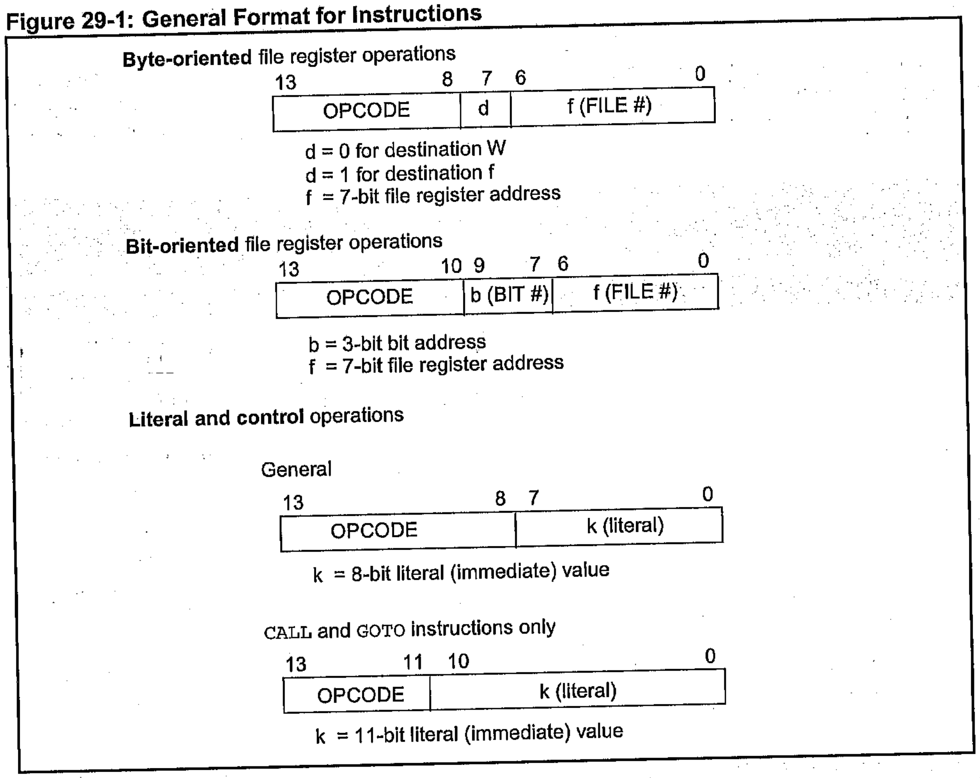
LOGIC GATES (HERE AND, OR, ADD, B-INVERT) SELECTED BY MUX'S DETERMINE OPERATIONS.

[Note SUBTRACTION, $A - B = A + (\overline{B} + 1)$

$= A +$ B-INVERT $+$ CARRY IN]

OP CODES CONTAIN BOTH ACTION AND ADDRESS / DATA INFORMATION

Figure 29-1: General Format for Instructions



Byte-oriented file register operations
13 8 7 6 0
OPCODE
d
f (FILE #)
Bit-oriented file register operations
13 10 9 7 6 0
OPCODE
b (BIT #)
f (FILE #)
Literal and control operations
General
13 8 7 0
OPCODE
k (literal)
CALL and GOTO instructions only
13 11 10 0
OPCODE
k (literal)

Byte-oriented file register operations

d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

Bit-oriented file register operations

b = 3-bit bit address
f = 7-bit file register address

Literal and control operations

General

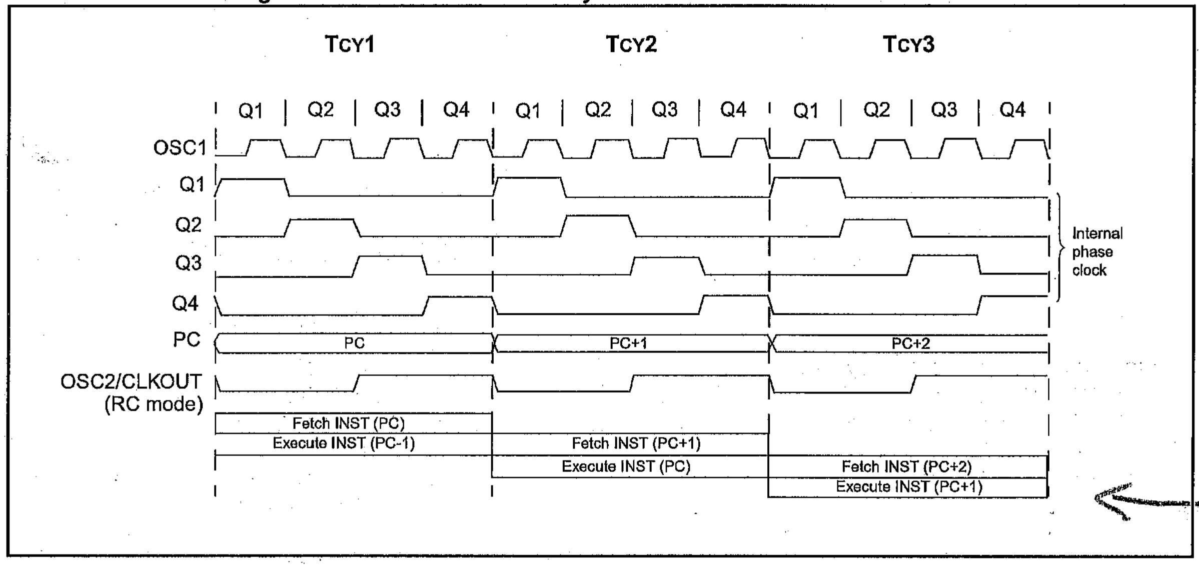
k = 8-bit literal (immediate) value

CALL and GOTO instructions only

k = 11-bit literal (immediate) value

INTERNAL CLOCKS SET UP 4 "Q-CYCLES" THAT AFFECT HOW PROCESSOR DECODES AND EXECUTES OPERATIONS

Figure 4-3: Clock/Instruction Cycle



WHILE 1 OPERATION HAPPENS ANOTHER GETS LOADED IN

INSTRUCTION SET – THE LANGUAGE OF PIC

TABLE 16-2: PIC16F87/88 INSTRUCTION SET

Mnemonic,	Operands	Description	Cycles	14-Bit Opcode				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1,2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	-	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	-	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	-	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Field	Description
`f`	Register file address (0x00 to 0x7F)
`W`	Working register (accumulator)
`b`	Bit address within an 8-bit file register (0 to 7)
`k`	Literal field, constant data or label (may be either an 8-bit or an 11-bit value)
`x`	Don't care (0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.

`d`	Destination select;
	d = 0: store result in W,
	d = 1: store result in file register f.

`dest`	Destination either the W register or the specified register file location

29.5 Instruction Descriptions

ADDLW Add Literal and W

Syntax:	[*label*] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	$(W) + k \rightarrow W$
Status Affected:	C, DC, Z
Encoding:	`11` | `111x` | `kkkk` | `kkkk`
Description:	The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process data	Write to W register

Example1 `ADDLW 0x15`

Before Instruction
W = 0x10
After Instruction
W = 0x25

Example 2 `ADDLW MYREG`

Before Instruction
W = 0x10
Address of `MYREG` † = 0x37
† `MYREG` is a symbol for a data memory location
After Instruction
W = 0x47

Example 3 `ADDLW HIGH (LU_TABLE)`

Before Instruction
W = 0x10
Address of `LU_TABLE` † = 0x9375
† `LU_TABLE` is a label for an address in program memory
After Instruction
W = 0xA3

Example 4 `ADDLW MYREG`

Before Instruction
W = 0x10
Address of `PCL` † = 0x02
† `PCL` is the symbol for the Program Counter low byte location
After Instruction
W = 0x12

BTFSC

Bit Test, Skip if Clear

Syntax: [*label*] BTFSC f,b

Operands: $0 \leq f \leq 127$
$0 \leq b \leq 7$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

01	10bb	bfff	ffff

Description: If bit 'b' in register 'f' is '0' then the next instruction is skipped.
If bit 'b' is '0' then the next instruction (fetched during the current instruction execution) is discarded, and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	No operation

If skip (2nd cycle):

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

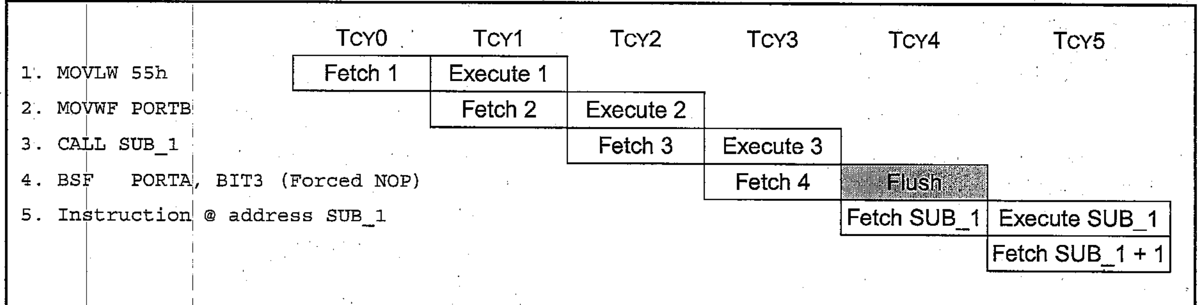
Example 1

```
HERE    BTFSC   FLAG, 4
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

Case 1: Before Instruction
PC = address HERE
FLAG= xxx0 xxxx
After Instruction
Since FLAG<4>= 0,
PC = address TRUE

Case 2: Before Instruction
PC = address HERE
FLAG= xxx1 xxxx
After Instruction
Since FLAG<4>=1,
PC = address FALSE

Example 4-1: Instruction Pipeline Flow

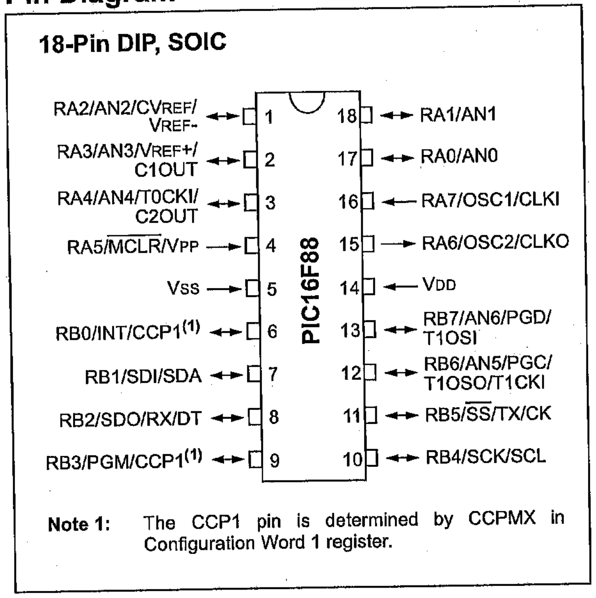


All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

INPUT / OUTPUT

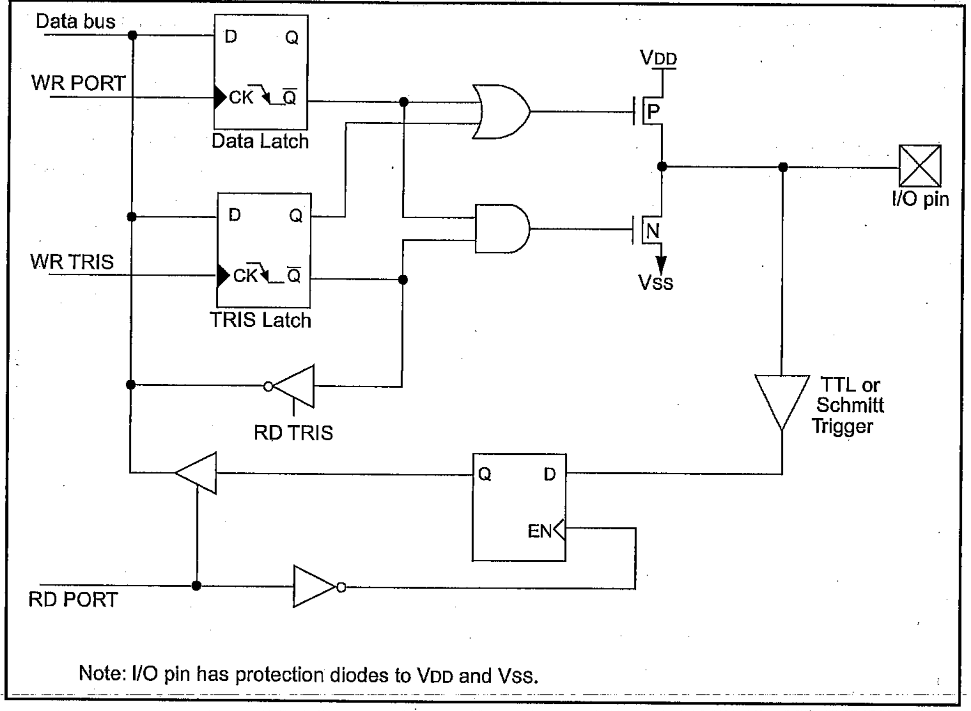
ALL PINS (EXCEPT POWER) DO MORE THAN ONE THING!!

Pin Diagram



Note 1: The CCP1 pin is determined by CCPMX in Configuration Word 1 register.

Figure 9-1: Typical I/O Port



Note: I/O pin has protection diodes to VDD and VSS.