The following three pages contain the 80386 instruction set. You do not need to submit these sheets when you finish.

Remember that:

- Most instructions can have at most one memory operand.
- Brackets [] around a register name, immediate, or combination of the two indicates an effective address. That address is in the data segment unless otherwise specified.
 - o Example: MOV AX, [10H] → contents of DS:10H moved to AX
- Parentheses around a logical address mean "the contents of memory at this address".
 - \circ Example: (DS:10H) \rightarrow the contents of memory at logical address DS:10H

Category	Instruction	Example	Meaning
	Move	MOV AX, BX	AX = BX
	Move & sign-extend	MOVSX EAX, DL	EAX = DL, sign-extended
			to 32 bits
	Move and zero-extend	MOVZX EAX, DL	EAX = DL, zero-extended
			to 32 bits
Data	Exchange	XCHG AX, BX	Swap contents of AX, BX
transfer	Load effective	LEA AX, [BX+SI+10H]	AX = BX + SI + 10H
	address		
	Load full pointer	LDS AX, [10H]	AX = (DS:10H)
			DS = (DS:12H)
		100 EDV [100H]	EDV - (DG : 100H)
		LSS EBX, [100H]	EBX = (DS:100H) SS = (DS:104H)
	Add	ADD AX, BX	AX = AX + BX
	Add with carry	ADC AX, BX	AX = AX + BX + CF
	Increment	INC [DI]	(DS:DI) = (DS:DI) + 1
	Subtract	SUB AX, [10H]	AX = AX - (DS:10H)
	Subtract with borrow	SBB AX, [10H]	AX = AX - (DS:10H) - CF
	Decrement	DEC CX	CX = CX - 1
	Negate (2's	NEG CX	CX = -CX
	complement)	NIG CX	
	Unsigned multiply	MUL BH	AX = BH * AL
	(all operands are non-	MUL CX	(DX,AX) = CX * AX
	negative, regardless	MUL DWORD PTR [10H]	(EDX,EAX) = (DS:10H) *
Arithmetic	of MSB value)		EAX
Antimetic	Signed multiply	IMUL BH	AX = BH * AL
	(all operands are	IMUL CX	(DX,AX) = CX * AX
	signed integers in 2's	IMUL DWORD PTR[10H]	(EDX,EAX) = (DS:10H) *
	complement form)		EAX
	Unsigned divide	DIV BH	AL = AX / BH (quotient)
			AH = AX % BH (remainder)
		DIV CX	AX = EAX / CX (quotient)
			DX = EAX % CX (remainder)
		DIV EBX	EAX = (EDX,EAX) / EBX (Q)
		DIA EDV	EAX = (EDX, EAX) / EBX (Q) EDX = (EDX, EAX) % EBX (R)
			EDV - (EDV'EVV) 0 EDV (K)

Category	Instruction	Example	Meaning
	Logical AND	AND AX, BX	AX = AX & BX
	Logical inclusive OR	OR AX, BX	AX = AX BX
Logical	Logical exclusive OR	XOR AX, BX	AX = AX ^ BX
	Logical NOT	NOT AX	AX = ~AX
	(1's complement)		
	Shift left	SHL AX, 7	$AX = AX \ll 7$
	Landard ability while	SAL AX, CX	AX = AX << CX
	Logical shift right	SHR AX, 7	AX = AX >> 7 (upper 7 bits = 0)
	(treat value as unsigned, shift in 0s)		(upper / bits = 0)
	Arithmetic shift right	SAR AX, 7	AX = AX >> 7
	(treat value as signed;	DAIC AX, /	(upper 7 bits = MSB of
Shift/rotate	maintain sign)		original value)
(NOTE: for	Rotate left	ROL AX, 7	AX = AX rotated left by 7
all		·	(lower 7 bits of AX =
instructions			upper 7 bits of original
except RCL/RCR,			value)
CF = last	Rotate right	ROR AX, 7	AX=AX rotated right by 7
bit shifted			(upper 7 bits of AX =
out)			lower 7 bits of original
,	Datata laft thursonle	DOI 3.7. 7	value)
	Rotate left through	RCL AX, 7	(CF,AX) rotated left by 7 (Treat CF & AX as 17-bit
	carry		value with CF as MSB)
	Rotate right through	RCR AX, 7	(AX,CX) rotated right by
	carry	resit ini, ,	7
			(Treat CF & AX as 17-b8t
			value with CF as LSB)
	Bit test	BT AX, 7	CF = Value of bit 7 of AX
	Bit test and reset	BTR AX, 7	CF = Value of bit 7 of AX
			Bit 7 of $AX = 0$
	Bit test and set	BTS AX, 7	CF = Value of bit 7 of AX
	Dit to at a mal	DEC AV 7	Bit 7 of AX = 1
	Bit test and	BTC AX, 7	CF = Value of bit 7 of AX Bit 7 of AX is flipped
	complement Bit scan forward	BSF DX, AX	DX = index of first non-
Bit test/	Dit Scall folward	BSI DX, AX	zero bit of AX, starting
scan			with bit 0
			ZF = 0 if AX = 0, 1
			otherwise
	Bit scan reverse	BSR DX, AX	DX = index of first non-
			zero bit of AX, starting
			with MSB
			ZF = 0 if $AX = 0$, 1
			otherwise

Category	Instruction	Example	Meaning			
	Clear carry flag	CLC	CF = 0			
	Set carry flag	STC	CF = 1			
	Complement carry	CMC	CF = ~CF			
	flag					
Floa	Clear interrupt flag	CLI	IF = 0			
Flag control	Set interrupt flag	STI	IF = 1			
CONTROL	Load AH with	LAHF	AH = FLAGS			
	contents of flags					
	register					
	Store contents of AH	SAHF	FLAGS = AH			
	in flags register		(Updates SF,ZF,AF,PF,CF)			
	Compare	CMP AX, BX	Subtract AX - BX			
Conditional			Updates flags			
tests	Byte set on condition	SETCC AH	AH = FF if condition true			
			AH = 0 if condition false			
	Unconditional jump	JMP label	Jump to label			
	Conditional jump	Jcc label	Jump to label if			
	1	7000 1 1 1	condition true			
	Loop	LOOP label	Decrement CX; jump to			
Jumps and	Lean if any al/none	LOODE label	label if CX != 0			
loops	Loop if equal/zero	LOOPE label	Decrement CX; jump to label if (CX != 0) &&			
		LOOPZ label	(ZF == 1)			
	Loop if not equal/zero	LOOPNE label	Decrement CX; jump to			
	Loop ii not equal/zero	LOOPNE label	label if (CX != 0) &&			
		HOOFNZ TADET	(ZF == 0)			
Subroutine-	Call subroutine	CALL label	Jump to label; save			
related			address of instruction			
instructions			after CALL			
	Return from	RET label	Return from subroutine			
	subroutine		(jump to saved address			
			from CALL)			
	Push	PUSH AX	SP = SP - 2			
			(SS:SP) = AX			
		PUSH EAX	SP = SP - 4			
			(SS:SP) = EAX			
	Pop	POP AX	AX = (SS:SP)			
			SP = SP + 2			
		DOD TAN	FAW (GG, GD)			
		POP EAX	EAX = (SS:SP)			
	Duch flass	DIGUE	SP = SP + 4			
	Push flags	PUSHF	Store flags on stack			
	Pop flags	POPF	Remove flags from stack			
	Push all registers	PUSHA	Store all general purpose registers on stack			
	Pop all registers	POPA	Remove general purpose			
		1	registers from stack			

The following three pages contain tables listing the PIC 16F684 instruction set, memory map, and block diagram.

Remember that, in the table below:

- f = a register file address
- W = the working register
- d = destination select: "F" for a file register, "W" for the working register
- b = bit position within an 8-bit file register
- k = literal field, constant data or label
- PC = the program counter
- C = the carry bit in the status register
- Z = the zero bit in the status register

TABLE 13-2: PIC16F684 INSTRUCTION SET

BYTE-ORIENTED FILE REGISTER OPERATIONS	Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status	Natas
ADDWF			Description		MSb			LSb	Affected	Notes
ANDWF		BYTE-ORIENTED FILE REGISTER OPERATIONS								
CLRF	ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C, DC, Z	1, 2
CLRW	ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1, 2
COMF	CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
DECF	CLRW	_	Clear W	1	00	0001	0xxx	XXXX	Z	
DECFSZ	COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1, 2
INCF	DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1, 2
INCFSZ	DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1, 2, 3
IORWF	INCF	f, d	Increment f	1	00	1010	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 C 1, 2 NOP No Operation 1 00 0000 0xx0 0x	INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1, 2, 3
MOVWF	IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1, 2
NOP	MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1, 2
RLF	MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
RRF	NOP	_	No Operation	1	00	0000	0xx0	0000		
RRF	RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	С	1, 2
SWAPF f, d Swap nibbles in f 1 00 1110 0fff fffff Z 1, 2	RRF	f, d		1	00	1100	dfff	ffff	С	1, 2
BIT-ORIENTED FILE REGISTER OPERATIONS	SUBWF	f, d	,	1	00	0010	dfff	ffff	C. DC. Z	1, 2
BIT-ORIENTED FILE REGISTER OPERATIONS	SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1, 2
BCF	XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1, 2
BSF			BIT-ORIENTED FILE REGIST	ER OPER	RATION	IS				
BTFSC f, b Bit Test f, Skip if Clear 1 (2) 01 10bb bfff ffff 3	BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1, 2
BTFSS f, b Bit Test f, Skip if Set 1 (2) 01 11bb bfff ffff 3	BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1, 2
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 TO, PD GOTO K Go to address 2 10 1kkk kkkk kkkk Z MOVLW 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 TO, PD SUBLW K Subtract W from literal 1 11 110x kkkk kkkk C, DC, Z	BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
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 Kkkk CLRWDT — Clear Watchdog Timer 1 00 0000 0110 0100 TO, PD TO, PD GOTO k Go to address 2 10 1kkk kkkk kkkk Lot, PD TO, PD GOTO Kkkk kkkk Kkkk Z MOVLW Inclusive OR literal with W 1 11 1000 kkkk kkkk Z AMD MOVLW Kkkk Kkkkk Kkkkk Kkkkk Kkkkk	BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
ANDLW k AND literal with W 1 11 1001 kkkk kkkk Z CALL k Call Subroutine 2 10 0kkk kkkk Z MOVLW k Inclusive OR literal with W 1 11 1000 kkkk kkkk Z MOVLW k Move literal to W 1 11 1000 kkkk K			LITERAL AND CONTROL	OPERAT	IONS					
CALL k Call Subroutine 2 10 0kkk kkkk kkkk CLRWDT — Clear Watchdog Timer 1 00 0000 0110 0100 TO, 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 kkkk RETFIE — Return from interrupt 2 00 0000 0000 1001 RETLW k Return with literal in W 2 11 01xx kkkk kkkk kkkk RETURN — Return from Subroutine 2 00 0000 0000 1000 100 0000 1001 TO, PD SUBLW K Subtract W from literal 1 11 110x kkkk kkkk C, DC, Z	ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C, DC, Z	
CLRWDT — Clear Watchdog Timer 1 00 0000 0110 0100 TO, PD GOTO k Go to address 2 10 1kkk kkkk kkkk kkkk kkkk kkkk kkkk Z MOVLW k Move literal to W 1 11 000x kkkk kkkk kkkk Rkkk kkkk Rkkk	ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
GOTO	CALL	k	Call Subroutine	2	10	0kkk	kkkk	kkkk		
IORLW k Inclusive OR literal with W 1 11 1000 kkkk kkkk Z	CLRWDT	_	Clear Watchdog Timer	1	00	0000	0110	0100	TO, PD	
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 TO, PD SUBLW k Subtract W from literal 1 11 110x kkkk kkkk C, DC, Z	GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
RETFIE	IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
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 TO, PD SUBLW K Subtract W from literal 1 11 110x kkkk kkkk C, DC, Z	MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
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 TO, PD SUBLW k Subtract W from literal 1 11 110x kkkk kkkk C, DC, Z	RETFIE	_	Return from interrupt	2	00	0000	0000	1001		
SLEEP	RETLW	k	•	2	11	01xx	kkkk	kkkk		
SUBLW k Subtract W from literal 1 11 110x kkkk kkkk C, DC, Z	RETURN	_	Return from Subroutine	2	00	0000	0000	1000		
SUBLW k Subtract W from literal 1 11 110x kkkk kkkk C, DC, Z	SLEEP	_	Go into Standby mode	1	00	0000	0110	0011	TO, PD	
	l	k	-	1	11					
XORLW k Exclusive OR literal with W 1 11 1010 kkkk kkkk Z	XORLW	k	Exclusive OR literal with W	1	11	1010			Z	

FIGURE 2-2: DATA MEMORY MAP OF THE PIC16F684

	File Address		File Address		
Indirect Addr.(1)	00h	Indirect Addr.(1)	80h		
TMR0	01h	OPTION_REG	81h		
PCL	02h	PCL PCL	82h		
STATUS	03h	STATUS	83h		
FSR	04h	FSR	84h		
PORTA	05h	TRISA	85h		
TORIA	06h	INIOA	86h		
PORTC	07h	TRISC	87h		
TORTO	08h	11100	88h		
	09h		89h		
PCLATH	0Ah	PCLATH	8Ah		
INTCON	0Bh	INTCON	8Bh		
PIR1		PIE1	_		
FINI	0Ch	FIET	8Ch		
TMD4I	0Dh	DOON	8Dh		
TMR1L	0Eh	PCON	8Eh		
TMR1H	0Fh	OSCCON	8Fh		
TICON	10h	OSCTUNE	90h		
TMR2	11h	ANSEL	91h		
T2CON	12h	PR2	92h		
CCPR1L	13h		93h		
CCPR1H	14h		94h		
CCP1CON	15h	WPUA	95h		
PWM1CON	16h	IOCA	96h		
ECCPAS	17h		97h		
WDTCON	18h		98h		
CMCON0	19h	VRCON	99h		
CMCON1	1Ah	EEDAT	9Ah		
	1Bh	EEADR	9Bh		
	1Ch	EECON1	9Ch		
	1Dh	EECON2 ⁽¹⁾	9Dh		
ADRESH	1Eh	ADRESL	9Eh		
ADCON0	1Fh	ADCON1	9Fh		
	20h	General	A0h		
		Purpose Registers			
General		32 Bytes	BFh		
Purpose					
Registers					
96 Bytes					
30 Dyies					
	6Fh				
	70		F0h		
	70 7Fh	Accesses 70h-7Fh	FUN FFh		
Bank 0	4 /FN	Bank 1	→ FFN		
Unimplemented data memory locations, read as '0'. Note 1: Not a physical register.					

INT 🛛 Configuration 13 **PORTA** Data Bus Program Counter Flash ĴĹ 2k X 14 Program RAM Memory 8-Level Stack 128 Bytes RA3 (13-Bit) File RA4 Registers Program 9 RAM Addr Bus Addr MUX Instruction Reg PORTC Indirect Direct Addr Addr RC0 RC1 FSR Reg RC2 STATUS Reg RC3 8 RC4 MUX Power-up Timer Instruction Oscillator Decode & Start-up Timer ALU Control Power-on Reset \times Timing Generation Watchdog OSC1/CLKIN W Reg \boxtimes Brown-out OSC2/CLKOUT Reset Internal Oscillator Block CCP1/P1A P1B P1C P1D \times X X \times \times T1G \bowtie VDD XT1CKI \boxtimes Timer0 Timer1 Timer2 **ECCP** T0CKI 2 Analog Comparators **EEDATA** Analog-To-Digital Converter and Reference 256 Bytes Data **EEPROM** EEADDR X \boxtimes \boxtimes \times \times \times X \times ANO AN1 AN2 AN3 AN4 AN5 AN6 AN7 C1IN- C1IN+ C1OUT C2IN- C2IN+ C2OUT

FIGURE 1-1: PIC16F684 BLOCK DIAGRAM

Source for all figures: "PIC 16F684 Data Sheet", Microchip Technology, Inc. http://ww1.microchip.com/downloads/en/DeviceDoc/41202F-print.pdf