How to Disassemble (aka Hack) MSP430 Machine Code

To decode an instruction:

1. Begin with a "**PC**" pointing to the first word of instruction in program memory. (Must be on word boundary.)
2. Retrieve first word (16 bits) of instruction from memory and increment PC by 2.
3. Lookup the corresponding instruction mnemonic using the opcode (most significant 4/6/9 bits) as an index into Table 3, 4, or 5. List the opcode mnemonic string.
4. If single or double operand instruction, append ".w" or ".b" to the opcode string (bit 6 = b/w, 0=word, 1=byte).
5. If double operand instruction (Table 5), decode and list source operand (Table 1).
6. If single or double operand instruction (Tables 3 and 5), decode and list destination operand (Table 2).
7. If jump instruction (Table 4), list the destination address given by sign extending the 10-bit PC offset (bits 0-9), multiplying by 2, and adding the result to the current PC.

To decode an operand:

1. To decode a source operand (single/double operand instructions):
   a. Decode the addressing mode from the "**As**" bits (00=register, 01=indexed, 10=indirect, or 11=indirect auto-increment) and source register from the "S-Reg" bits.
   b. If "**@R2**", "**@R2+**", "**R3**", "**x(R3)**", "**@R3**", or "**@R3+**", list appropriate constant preceded by pound sign (ie #1).
   c. Else if "**x(R0)**", change to symbolic mode, retrieve index (next code word), add index word to PC, increment PC, and list that address as operand (ie. 0x8023).
   d. Else if "***x*(R2)**", change to absolute mode, retrieve address (next code word), increment PC, and list address preceded by an ampersand symbol (ie. &addr).
   e. Else if "**@PC+**", change to immediate mode, retrieve immediate value (next code word), increment PC, and list immediate value preceded by the pound symbol (ie. #100).
   f. Else if **register mode**, list register (ie. R*n*).
   g. Else if **indexed mode**, retrieve index (next code word), increment PC, and list index followed by the register in parentheses (ie. 0x0200(R4)).
   h. Else if **indirect mode**, list the register preceded by an @ symbol (ie. @R4).
   i. Else **indirect auto-increment mode**, list the register preceded by an @ symbol and followed by a plus symbol (ie. @R4+).
2. To decode a destination operand (double operand instructions only), use the "**Ad**" bit and the destination register bits. Follow the same steps as for the source operand (except there will only be register and indexed modes – no constants, immediate, or indirect modes).

Table 1. Source Addressing Modes (As)

Address Mode	*As	Registers	Syntax	Operation
Register	**00**	**R0-R2, R4-R15**	**R*n***	**Register Contents.**
0	**00**	**R3**	**#0**	**0 Constant**
Symbolic	**01**	**R0**	***addr***	**(PC+next word) points to operand. (*x*(PC))**
Indexed	**01**	**R1, R4-R15**	***x*(R*n*)**	**(R*n*+*x*) points to operand. *x* is next code word.**
Absolute	**01**	**R2**	**&*addr***	**Next code word is the absolute address. (*x*(SR))**
+1	**01**	**R3**	**#1**	**+1 Constant**
Indirect	**10**	**R0-R1,R4-R15**	**@R*n***	**R*n* points to operand.**
+4	**10**	**R2**	**#4**	**+4 Constant**
+2	**10**	**R3**	**#2**	**+2 Constant**
Immediate	**11**	**R0**	**#*N***	**Next word is the constant *N*. (@PC+)**
Indirect auto-inc	**11**	**R1,R4-R15**	**@R*n*+**	**R*n* points to operand, R*n* is incremented (1 or 2).**
+8	**11**	**R2**	**#8**	**+8 Constant**
-1	**11**	**R3**	**#-1**	**-1 Constant**

***Bits 4 and 5 in Single (Table 3) and Double (Table 5) Operand Instructions**

Table 2. Destination Addressing Modes (Ad)

Address Mode	*Ad	Registers	Syntax	Operation
Register	**0**	**R0-R2, R4-R15**	**R*n***	**Register Contents.**
0	**0**	**R3**	**#0**	**Bit bucket**
Symbolic	**1**	**R0**	***addr***	**(PC+next word) points to operand. (*x*(PC))**
Indexed	**1**	**R1, R4-R15**	***x*(R*n*)**	**(R*n*+*x*) points to operand. *x* is next code word.**
Absolute	**1**	**R2**	**&*addr***	**Next code word is the absolute address. (*x*(SR))**

***Bit 7 in (Table 5) Operand Instructions**

Table 3. Single Operand Instructions

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
9-bit Opcode									b/w	As		D/S Register			

Mnemonic	Opcode	V	N	Z	C	Operation	Description
RRC	0 0 0 1 0 0 0 0 0	•	•	•	•	C→MSB→…LSB→C	Roll dst right through C
SWPB	0 0 0 1 0 0 0 0 1	–	–	–	–	Swap bytes	Swap bytes
RRA	0 0 0 1 0 0 0 1 0	0	•	•	•	MSB→MSB→…LSB→C	Roll destination right
SXT	0 0 0 1 0 0 0 1 1	0	•	•	z	bit 7→bit 8…bit 15	Sign extend destination
PUSH	0 0 0 1 0 0 1 0 0	–	–	–	–	SP-2→SP, src→@SP	Push source on stack
CALL	0 0 0 1 0 0 1 0 1	–	–	–	–	SP-2→SP, PC+2→@SP, dst→PC	Subroutine call
RETI	0 0 0 1 0 0 1 1 0	•	•	•	•	@SP+→SR, @SP+→PC	Return from interrupt

Table 4. Jump Instructions

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
6-bit Opcode						10-bit, 2's complement PC Offset									

Mnemonic	Opcode	V	N	Z	C	Description
JNZ/JNE	0 0 1 0 0 0	–	–	0	–	Jump if not equal
JZ/JEQ	0 0 1 0 0 1	–	–	1	–	Jump if equal
JNC/JLO	0 0 1 0 1 0	–	–	–	0	Jump if carry flag equal to zero
JC/JHS	0 0 1 0 1 1	–	–	–	1	Jump if carry flag equal to one
JN	0 0 1 1 0 0	–	1	–	–	Jump if negative (N = 1)
JGE	0 0 1 1 0 1	•	•	–	–	Jump if greater than or equal (N = V)
JL	0 0 1 1 1 0	•	•	–	–	Jump if lower (N ≠ V)
JMP	0 0 1 1 1 1	–	–	–	–	Unconditional jump

Table 5. Double Operand Instructions

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4-bit Opcode				Source Register				Ad	b/w	As		Destination Register			

Mnemonic	Opcode	V	N	Z	C	Operation	Description
MOV	0 1 0 0	–	–	–	–	src→dst	Move source to destination
ADD	0 1 0 1	•	•	•	•	src+dst→dst	Add source to destination
ADDC	0 1 1 0	•	•	•	•	src+dst+C→dst	Add src and C to dst
SUBC	0 1 1 1	•	•	•	•	dst+.not.src+C→dst	Subtract src and NOT C from dst
SUB	1 0 0 0	•	•	•	•	dst+.not.src+1→dst	Subtract source from destination
CMP	1 0 0 1	•	•	•	•	dst-src	Compare source to destination
DADD	1 0 1 0	•	•	•	•	src+dst+C→dst(dec)	Decimal add src and C to dst
BIT	1 0 1 1	0	•	•	z	src.and.dst	Test bits in destination
BIC	1 1 0 0	–	–	–	–	.not.src.and.dst→dst	Clear bits in destination
BIS	1 1 0 1	–	–	–	–	src.or.dst→dst	Set bits in destination
XOR	1 1 1 0	•	•	•	z	src.xor.dst→dst	XOR source with destination
AND	1 1 1 1	0	•	•	z	src.and.dst→dst	AND source with destination

Status Register: • = bit affected, – = bit not affected, 0 = cleared, 1 = set, z = same as Z

Table 6. Source Operands Using Status (R2) and Constant Generator (R3) Registers

Register	As	Addr Mode	Syntax	Constant	Remarks
R2	00	Register	–	–	Register mode
R2	01	*x*(R2)	*addr*	(0)	Absolute address mode, next word contains address
R2	10	@R2	#4	0x0004	+4
R2	11	@R2+	#8	0x0008	+8
R3	00	R3	#0	0x0000	0
R3	01	*x*(R3)	#1	0x0001	+1, No extension word
R3	10	@R3	#2	0x0002	+2
R3	11	@R3+	#-1	0xFFFF	-1