

-- Arquitetura e Organização de Computadores
-- Arquivo: instruções.txt
-- Anderson Cottica
-- Erik Ryuichi Yamamoto
-- Data entrega: 26/10/17

Para o funcionamento de nosso microprocessador foi necessário implementar algumas instruções e sinais que são apresentados a seguir:

1 - clk – clock	-- sinal de clock do sistema
2 - rst - reset	-- reset do sistema
3 - estado_s0	-- para máquina de estados, estado 0
4 - estado_s1	-- para máquina de estados, estado 1
5 - estado_s2	-- para máquina de estados, estado 2
6 - mov_s	-- habilita a instrução MOV
7 - add_s	-- habilita a instrução ADD
8 - sub_s	--habilita a instrução SUB
9 - cmp_s	--habilita a instrução CMP
10 - ld_s	-- habilita a inst. LD, manipulação de valores na memória
11 - st_s	-- habilita a instrução ST
12 - reg1 ~ reg6	-- registradores
13 - pc_s	-- saída do PC
14 - reg_dest_s	-- registrador destino
15 - reg_fonte_s	-- registrador fonte

Os opcodes implementados em nosso processador tem o seguinte formato para 15 bits da ROM:

Para o TIPO R:

MOV const, Rd	-- 0100 CCCCCCCC DDD
MOV Rs, Rd	-- 0100 SSS 00000 DDD
ADD const, Rd	-- 0101 CCCCCCCC DDD
ADD Rs, Rd	-- 0101 SSS 00000 DDD
SUB const, Rd	-- 1000 CCCCCCCC DDD
SUB Rs, Rd	-- 1000 SSS 00000 DDD
CMP const, Rd	-- 1001 CCCCCCCC DDD
CMP Rs, Rd	-- 1001 SSS 00000 DDD
ST Rs, @Rd	-- 1100 SSS 00000 DDD
LD Rs, @Rd	-- 1101 SSS 00000 DDD

A operação MOV trata da movimentação entre valores de registradores ou entre registrador e um valor constante. Da mesma forma, ADD faz uma operação aritmética entre registradores ou entre registrador e uma constante. SUB trata-se da subtração entre valores de registradores ou entre registrador e uma constante. CMP é a comparação entre dois valores de registradores para definição se os valores são iguais ou diferentes a fim de realizar o salto para determinada parte do programa. ST e LD são instruções para leitura e escrita de valores na memória.

Para o TIPO J:

JMP endereço -- 001111 EEEEEEEEE

NOP -- 0000000000000000

O tipo de instrução J representa os saltos do sistema.

OBS.:

C - valor da constante em binário

S - Valor do registrador fonte em binário

D - Valor do registrador de destino em binário

CMP - compara registrador de destino com uma constante ou com um registrador fonte e seta flag para o caso em que o Registrador de destino for maior ou igual, ou zero quando menor.

JMP - pula incondicionalmente para o endereço solicitado.

ST - carrega o valor de registrador fonte no endereço dado pelo registrador de destino.

LD - carrega o valor no endereço dado do registrador de destino, no registrador fonte.

A tabela a seguir apresenta as instruções destacadas a serem implementadas e seus respectivos opcodes:

MSP430 instruction set

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Instruction	
0	0	0	1	0	0	opcode		B/W	As	register		Single-operand arithmetic					
0	0	0	1	0	0	0	0	0	B/W	As	register		RRC Rotate right (1 bit) through carry				
0	0	0	1	0	0	0	0	1	0	As	register		SWPB Swap bytes				
0	0	0	1	0	0	0	1	0	B/W	As	register		RRA Rotate right (1 bit) arithmetic				
0	0	0	1	0	0	0	1	1	0	As	register		SXT Sign extend byte to word				
0	0	0	1	0	0	1	0	0	B/W	As	register		PUSH Push value onto stack				
0	0	0	1	0	0	1	0	1	0	As	register		CALL Subroutine call; push PC and move source to PC				
0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	RETI Return from interrupt; pop SR then pop PC	
0	0	1	condition		10-bit signed offset							Conditional jump; PC = PC + 2×offset					
0	0	1	0	0	0	10-bit signed offset							JNE/JNZ Jump if not equal/zero				
0	0	1	0	0	1	10-bit signed offset							JEQ/JZ Jump if equal/zero				
0	0	1	0	1	0	10-bit signed offset							JNC/JLO Jump if no carry/lower				
0	0	1	0	1	1	10-bit signed offset							JC/JHS Jump if carry/higher or same				
0	0	1	1	0	0	10-bit signed offset							JN Jump if negative				
0	0	1	1	0	1	10-bit signed offset							JGE Jump if greater or equal				
0	0	1	1	1	0	10-bit signed offset							JL Jump if less				
0	0	1	1	1	1	10-bit signed offset							JMP Jump (unconditionally)				
opcode			source			Ad	B/W	As	destination		Two-operand arithmetic						
0	1	0	0	source		Ad	B/W	As	destination		MOV Move source to destination						
0	1	0	1	source		Ad	B/W	As	destination		ADD Add source to destination						
0	1	1	0	source		Ad	B/W	As	destination		ADDC Add source and carry to destination						
0	1	1	1	source		Ad	B/W	As	destination		SUBC Subtract source from destination (with carry)						
1	0	0	0	source		Ad	B/W	As	destination		SUB Subtract source from destination						
1	0	0	1	source		Ad	B/W	As	destination		CMP Compare (pretend to subtract) source from destination						
1	0	1	0	source		Ad	B/W	As	destination		DADD Decimal add source to destination (with carry)						
1	0	1	1	source		Ad	B/W	As	destination		BIT Test bits of source AND destination						
1	1	0	0	source		Ad	B/W	As	destination		BIC Bit clear (dest &= ~src)						
1	1	0	1	source		Ad	B/W	As	destination		BIS Bit set (logical OR)						
1	1	1	0	source		Ad	B/W	As	destination		XOR Exclusive or source with destination						
1	1	1	1	source		Ad	B/W	As	destination		AND Logical AND source with destination (dest &= src)						

Referência utilizada: <https://en.wikipedia.org/wiki/TI_MSP430>