JSR — Jump to Subroutine — JSR

Operation:

PC ⇐ (PC) + $0003 (for EXTended or INDexed, Y addressing)
or:
PC ⇐ (PC) + $0002 (for DIRect or INDexed, X addressing)
⇓ (PCL) Push low-order return address onto stack
SP ⇐ (SP) – $0001
⇓ (PCH) Push high-order return address onto stack
SP ⇐ (SP) – $0001
PC ⇐ Effective Addr Load start address or requested subroutine

Description: The program counter is incremented by three or by two, depending on the addressing mode, and is then pushed onto the stack, eight bits at a time, least significant byte first. The stack pointer points to the next empty location in the stack. A jump occurs to the instruction stored at the effective address. The effective address is obtained according to the rules for EXTended, DIRect, or INDexed addressing.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: JSR (opr)

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	JSR (DIR)			JSR (EXT)			JSR (IND,X)			JSR (IND,Y)		
	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W	Addr	Data	R/W
1	OP	9D	1	OP	BD	1	OP	AD	1	OP	18	1
2	OP + 1	dd	1	OP + 1	hh	1	OP + 1	ff	1	OP + 1	AD	1
3	00dd	(00dd)	1	OP + 2	ll	1	FFFF	—	1	OP + 2	ff	1
4	SP	Rtn lo	0	hhll	(hhll)	1	X + ff	(X + ff)	1	FFFF	—	1
5	SP – 1	Rtn hi	0	SP	Rtn lo	0	SP	Rtn lo	0	Y + ff	(Y + ff)	1
6				SP – 1	Rtn hi	0	SP – 1	Rtn hi	0	SP	Rtn lo	0
7										SP – 1	Rtn hi	0

For More Information On This Product,
Go to: www.freescale.com

RTS — Return from Subroutine — RTS

Operation:

SP ⇐ (SP) + $0001, ⇑ (PCH)
SP ⇐ (SP) + $0001, ⇑ (PCL)

Description: The stack pointer is incremented by one. The contents of the byte of memory, at the address now contained in the stack pointer, are loaded into the high-order eight bits of the program counter. The stack pointer is again incremented by one. The contents of the byte of memory, at the address now contained in the stack pointer, are loaded into the low-order eight bits of the program counter.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Form: RTS

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	RTS (INH)		
	Addr	Data	R/W
1	OP	39	1
2	OP + 1	—	1
3	SP	—	1
4	SP + 1	Rtn hi	1
5	SP + 2	Rtn lo	1

For More Information On This Product,
Go to: www.freescale.com

PSH — Push Data onto Stack — PSH

Operation: ⇓ ACCX, SP ⇐ (SP) – $0001

Description: The contents of ACCX are stored on the stack at the address contained in the stack pointer. The stack pointer is then decremented.

Push instructions are commonly used to save the contents of one or more CPU registers at the start of a subroutine. Just before returning from the subroutine, corresponding pull instructions are used to restore the saved CPU registers so the subroutine will appear not to have affected these registers.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Forms: PSHA; PSHB

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	PSHA (INH)			PSHB (INH)		
	Addr	Data	R/W	Addr	Data	R/W
1	OP	36	1	OP	37	1
2	OP + 1	—	1	OP + 1	—	1
3	SP	(A)	0	SP	(B)	0

For More Information On This Product,
Go to: www.freescale.com

PUL — Pull Data from Stack — PUL

Operation: SP ⇐ (SP) + $0001, ⇑ (ACCX)

Description: The stack pointer is incremented. The ACCX is then loaded from the stack at the address contained in the stack pointer.

Push instructions are commonly used to save the contents of one or more CPU registers at the start of a subroutine. Just before returning from the subroutine, corresponding pull instructions are used to restore the saved CPU registers so the subroutine will appear not to have affected these registers.

Condition Codes and Boolean Formulae:

S	X	H	I	N	Z	V	C
—	—	—	—	—	—	—	—

None affected

Source Forms: PULA; PULB

Addressing Modes, Machine Code, and Cycle-by-Cycle Execution:

Cycle	PULA (INH)			PULB (INH)		
	Addr	Data	R/W	Addr	Data	R/W
1	OP	32	1	OP	33	1
2	OP + 1	—	1	OP + 1	—	1
3	SP	—	1	SP	—	1
4	SP + 1	get A	1	SP + 1	get B	1

For More Information On This Product,
Go to: www.freescale.com