VIRTUAL16 Instruction Set

	1 st E	3yte	2 nd Byte		3 rd Byte		
	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	See
RET	0	0	0	0	-	-	1
MOV Rs, Rd	0	1	Rs	Rd	-	-	
MOV @Rs, Rd	0	2	Rs	Rd	-	-	8
MOV Rs, @Rd	0	3	Rs	Rd	-	-	8
MOV +@Rs, Rd	0	4	Rs	Rd	-	-	8
MOV Rs, +@Rd	0	5	Rs	Rd	-	-	8
PUSH Rd	0	6	Rd	Χ	-	-	6
POP Rd	0	7	Rd	X	-	-	6
CMP Rs, Rd	0	8	Rs	Rd	-	-	10
SWAP Rs, Rd	0	9	Rs	Rd	-	-	2
AND Rs, Rd	0	Α	Rs	Rd	-	-	
OR Rs, Rd	0	В	Rs	Rd	-	-	
XOR Rs, Rd	0	С	Rs	Rd	-	-	
JSR ADDR	0	D	ADDR[7:4]	ADDR[3:0]	ADDR[15:12]	ADDR[11:8]	3
RTS	0	Е	-	-	-	-	
JMP ADDR	0	F	ADDR[7:4]	ADDR[3:0]	ADDR[15:12]	ADDR[11:8]	3
	451 0		and Dutto		Ord Purto		
	1 St .	Durto	and	D. do	ard I	Dusta	
		Byte		Byte		Byte	Soo
INC Dd. #COUNT	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	3 rd I Hi Nb (BIT7:4)	Byte Lo Nb (BIT3:0)	See
INC Rd, #COUNT	Hi Nb (BIT7:4) 1	Lo Nb (BIT3:0) 0	Hi Nb (BIT7:4) Rd	Lo Nb (BIT3:0) COUNT		•	4
DEC Rd, #COUNT	Hi Nb (BIT7:4) 1 1	Lo Nb (BIT3:0) 0 1	Hi Nb (BIT7:4) Rd Rd	Lo Nb (BIT3:0) COUNT COUNT		•	
DEC Rd, #COUNT ADD Rs, Rd	Hi Nb (BIT7:4) 1 1 1	Lo Nb (BIT3:0) 0 1 2	Hi Nb (BIT7:4) Rd Rd Rs	Lo Nb (BIT3:0) COUNT COUNT Rd		•	4
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd	Hi Nb (BIT7:4) 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3	Hi Nb (BIT7:4) Rd Rd Rs Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd	Hi Nb (BIT7:4) - - - -	•	4
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd	Hi Nb (BIT7:4) 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4	Hi Nb (BIT7:4) Rd Rd Rs Rs Rs Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd		•	4
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd	Hi Nb (BIT7:4) 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5	Hi Nb (BIT7:4) Rd Rd Rs Rs Rs Rs Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	4 4
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd SMUL Rs, Rd	Hi Nb (BIT7:4) 1 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5 6	Hi Nb (BIT7:4) Rd Rd Rs Rs Rs Rs Rs Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd Rd Rd Rd	Hi Nb (BIT7:4) - - - -	•	10
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd SMUL Rs, Rd UMUL Rs, Rd	Hi Nb (BIT7:4) 1 1 1 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5 6 7	Hi Nb (BIT7:4) Rd Rd Rs Rs Rs Rs Rs Rs Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd Rd Rd Rd Rd	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	10 10
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd SMUL Rs, Rd UMUL Rs, Rd RJSR ADDR	Hi Nb (BIT7:4) 1 1 1 1 1 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5 6 7 8	Hi Nb (BIT7:4) Rd Rd Rs Rs Rs Rs Rs Rs Rs ADDR[7:4]	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd Rd Rd Rd ADDR[3:0]	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	10 10 7
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd SMUL Rs, Rd UMUL Rs, Rd RJSR ADDR ASR Rd, #COUNT	Hi Nb (BIT7:4) 1 1 1 1 1 1 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5 6 7 8 9	Hi Nb (BIT7:4) Rd Rd Rs Rs Rs Rs Rs Rs ADDR[7:4]	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd Rd Rd ADDR[3:0] COUNT	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	10 10 7 4
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd SMUL Rs, Rd UMUL Rs, Rd RJSR ADDR ASR Rd, #COUNT LSL Rd, #COUNT	Hi Nb (BIT7:4) 1 1 1 1 1 1 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5 6 7 8 9 A	Hi Nb (BIT7:4) Rd Rd Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd Rd ADDR[3:0] COUNT COUNT	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	10 10 7 4 4
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd SMUL Rs, Rd UMUL Rs, Rd RJSR ADDR ASR Rd, #COUNT LSL Rd, #COUNT LSR Rd, #COUNT	Hi Nb (BIT7:4) 1 1 1 1 1 1 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5 6 7 8 9 A B	Hi Nb (BIT7:4) Rd Rd Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd Rd ADDR[3:0] COUNT COUNT COUNT	Hi Nb (BIT7:4) ADDR[15:12]	Lo Nb (BIT3:0)	10 10 7 4 4 4
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd SMUL Rs, Rd UMUL Rs, Rd RJSR ADDR ASR Rd, #COUNT LSL Rd, #COUNT ROL Rd, #COUNT	Hi Nb (BIT7:4) 1 1 1 1 1 1 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5 6 7 8 9 A B C	Hi Nb (BIT7:4) Rd Rd Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd Rd ADDR[3:0] COUNT COUNT COUNT COUNT	Hi Nb (BIT7:4) ADDR[15:12]	Lo Nb (BIT3:0)	10 10 7 4 4 4 4, 9
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd SMUL Rs, Rd UMUL Rs, Rd RJSR ADDR ASR Rd, #COUNT LSL Rd, #COUNT LSR Rd, #COUNT ROL Rd, #COUNT RLC Rd, #COUNT	Hi Nb (BIT7:4) 1 1 1 1 1 1 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5 6 7 8 9 A B C D	Hi Nb (BIT7:4) Rd Rd Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd Rd ADDR[3:0] COUNT COUNT COUNT COUNT COUNT COUNT	Hi Nb (BIT7:4) ADDR[15:12]	Lo Nb (BIT3:0)	10 10 7 4 4 4, 9 4, 9
DEC Rd, #COUNT ADD Rs, Rd ADC Rs, Rd SUB Rs, Rd SBC Rs, Rd SMUL Rs, Rd UMUL Rs, Rd RJSR ADDR ASR Rd, #COUNT LSL Rd, #COUNT ROL Rd, #COUNT	Hi Nb (BIT7:4) 1 1 1 1 1 1 1 1 1 1 1 1 1	Lo Nb (BIT3:0) 0 1 2 3 4 5 6 7 8 9 A B C	Hi Nb (BIT7:4) Rd Rd Rs	Lo Nb (BIT3:0) COUNT COUNT Rd Rd Rd Rd Rd Rd ADDR[3:0] COUNT COUNT COUNT COUNT	Hi Nb (BIT7:4) ADDR[15:12]	Lo Nb (BIT3:0)	10 10 7 4 4 4 4, 9

VIRTUAL16 Instruction Set (continued)

	1 st Byte		2 nd Byte		3 rd Byte		
	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	Hi Nb (BIT7:4)	Lo Nb (BIT3:0)	See
MOVL Rd, #VAL	2	Rd	VAL[7:4]	VAL[3:0]	-	-	
MOVH Rd, #VAL	3	Rd	VAL[7:4]	VAL[3:0]	-	-	
MOV Rd, #VAL	4	Rd	VAL[7:4]	VAL[3:0]	VAL[15:12]	VAL[11:8]	
BPL Rs, OFFSET	5	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BMI Rs, OFFSET	6	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BEQ Rs, OFFSET	7	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BNE Rs, OFFSET	8	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BM1 Rs, OFFSET	9	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BNM1 Rs, OFFSET	Α	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BRA OFFSET	В	X	OFFSET[7:4]	OFFSET[3:0]	-	-	5, 6
BCS OFFSET	С	X	OFFSET[7:4]	OFFSET[3:0]	-	-	5, 6
BCC OFFSET	D	X	OFFSET[7:4]	OFFSET[3:0]	-	-	5, 6
MOV ADDR, Rd	E	Rd	ADDR[7:4]	ADDR[3:0]	ADDR[15:12]	ADDR[11:8]	
MOV Rs, ADDR	F	Rs	ADDR[7:4]	ADDR[3:0]	ADDR[15:12]	ADDR[11:8]	

- 1: RET instruction must be last instruction always
- 2: SWAP instruction have 2 modes. If source and destination register is different it swaps register contents. If source and destination register is same it swaps LSB and MSB of register
- 3: JSR and JMP calculates destination address automatically. Assemble your VIRTUAL16 code starting from 0x0000 address for proper calculation
- 4: #COUNT must be "real count 1". If #COUNT == 0 VIRTUAL16 calculates it as 1, if #COUNT == 15 VIRTUAL16 calculates it as 16 etc.
- 5: OFFSET formula is = New Address Current Address 2. Same as 6502. Except BRA, BCS and BCC instructions you must give Register to test
- 6: x means don't matter. You can give any value to this nibbles
- 7: With this instruction you can execute 6502 subroutine without returning from VIRTUAL16 mode. Just give it 6502 subroutine address and magic will happen. Your subroutine must end with RTS instruction
- 8: @ means register holds pointer. +@ means register holds pointer and after execution pointer will incremented by 2. These instructions destroy BITO of pointer. It means you cannot access odd locations in memory
- 9: Difference between ROL and RLC is ROL ignores carry for first rotate but RLC don't. Same for ROR and RRC.
- 10: SMUL and UMUL instructions are 16x16=32-bit and result stored at R11 (LSW) and R12 (MSW). CMP instruction result stored at R13. You're free to use these registers but if you use SMUL, UMUL and CMP instructions register contents are replaced with result