VIRTUAL16 Instruction Set

	1st Byte		2nd Byte		3rd 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	-	-	-	-	1
RTS	0	1	-	-	-	-	
CLC	0	2	-	-	-	-	
SEC	0	3	-	-	-	-	
MOV Rs, Rd	0	4	Rs	Rd	-	-	8
MOV @Rs, Rd	0	5	Rs	Rd	-	-	8
MOV Rs, @Rd	0	6	Rs	Rd	-	-	8
MOV +@Rs, Rd	0	7	Rs	Rd	-	-	8
MOV Rs, +@Rd	0	8	Rs	Rd	-	-	8
MOV Rs.H, Rd.H	0	9	Rs	Rd	Rd(MSB) = Rs(MSB)		
MOV Rs.H, Rd.L	0	A	Rs	Rd	Rd(MSB) = Rs(LSB)		
MOV Rs.L, Rd.H	0	B	Rs	Rd	Rd(LSB) = Rs(MSB)		
MOV Rs.L, Rd.L	0	C	Rs	Rd	Rd(LSB) = Rs(LSB)		
SWAP Rs, Rd	0	D	Rs	Rd	-	-	2
AND Rd, Rs	0	E	Rd	Rs	Rd = Rd & Rs		
OR Rd, Rs	0	F	Rd	Rs	Rd = Rd | Rs		
	1st Byte		2nd Byte		3rd 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
XOR Rd, Rs	1	0	Rd	Rs	Rd = Rd ^ Rs		
ADD Rd, Rs	1	1	Rd	Rs	Rd = Rd + Rs		
ADC Rd, Rs	1	2	Rd	Rs	Rd = Rd + Rs + Carry		
SUB Rd, Rs	1	3	Rd	Rs	Rd = Rd – Rs		
SBC Rd, Rs	1	4	Rd	Rs	Rd = Rd – Rs - !Carry		
SMUL Rd, Rs	1	5	Rd	Rs	R12(LSW):R13(MSW) = Rs * Rd		10
UMUL Rd, Rs	1	6	Rd	Rs	R12(LSW):R13(MSW) = Rs * Rd		10
CMP Rd, Rs	1	7	Rd	Rs	R13 = Rd – Rs		10
INC Rd, #COUNT	1	8	Rd	COUNT	-	-	4
DEC Rd, #COUNT	1	9	Rd	COUNT	-	-	4
ASR Rd, #COUNT	1	A	Rd	COUNT	-	-	4
LSL Rd, #COUNT	1	B	Rd	COUNT	-	-	4
LSR Rd, #COUNT	1	C	Rd	COUNT	-	-	4
ROL Rd, #COUNT	1	D	Rd	COUNT	-	-	4, 9
RLC Rd, #COUNT	1	E	Rd	COUNT	-	-	4, 9
ROR Rd, #COUNT	1	F	Rd	COUNT	-	-	4, 9

VIRTUAL16 Instruction Set (continued)

	1st Byte		2nd Byte		3rd 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
RRC Rd, #COUNT	2	0	Rd	COUNT	-	-	4, 9
JMPR Rd	2	1	Rd	x	-	-	
JSR ADDR	2	2	ADDR[7:4]	ADDR[3:0]	ADDR[15:12]	ADDR[11:8]	3
NJSR ADDR	2	3	ADDR[7:4]	ADDR[3:0]	ADDR[15:12]	ADDR[11:8]	7
JMP ADDR	2	4	ADDR[7:4]	ADDR[3:0]	ADDR[15:12]	ADDR[11:8]	3
BCC OFFSET	2	5	OFFSET[7:4]	OFFSET[3:0]	-	-	5, 6
BCS OFFSET	2	6	OFFSET[7:4]	OFFSET[3:0]	-	-	5, 6
BRA OFFSET	2	7	OFFSET[7:4]	OFFSET[3:0]	-	-	5, 6
NOP	2	8	-	-	-	-	
NOP	2	9	-	-	-	-	
NOP	2	A	-	-	-	-	
NOP	2	B	-	-	-	-	
NOP	2	C	-	-	-	-	
NOP	2	D	-	-	-	-	
NOP	2	E	-	-	-	-	
NOP	2	F	-	-	-	-	
	1st Byte		2nd Byte		3rd 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
PUSH Rd	3	Rd	-	-	-	-	6
POP Rd	4	Rd	-	-	-	-	6
MOV #VAL, Rd.L	5	Rd	VAL[7:4]	VAL[3:0]	-	-	
MOV #VAL, Rd.H	6	Rd	VAL[7:4]	VAL[3:0]	-	-	
MOV #VAL, Rd	7	Rd	VAL[7:4]	VAL[3:0]	VAL[15:12]	VAL[11:8]	
MOV ADDR, Rd	8	Rd	ADDR[7:4]	ADDR[3:0]	ADDR[15:12]	ADDR[11:8]	
MOV Rs, ADDR	9	Rs	ADDR[7:4]	ADDR[3:0]	ADDR[15:12]	ADDR[11:8]	
BNM1 Rs, OFFSET	A	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BM1 Rs, OFFSET	B	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BMI Rs, OFFSET	C	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BPL Rs, OFFSET	D	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BNE Rs, OFFSET	E	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5
BEQ Rs, OFFSET	F	Rs	OFFSET[7:4]	OFFSET[3:0]	-	-	5

VIRTUAL16 Instruction Set (continued)

1: Last instruction must be RET

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. If #COUNT not given to customasm assembler it will increment or decrement by 1

5: OFFSET formula is = New Address – Current Address – 2. Same as 6502. Except BRA, BCS and BCC instructions, you must give register to test. If no register given to customasm assembler it will test Compare Register (R13)

6: x means don’t matter. You can give any value to this nibbles

7: With this instruction you can execute native 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 BIT0 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.