Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS 081
FoE&T Semester: VI Session: 2021-22
Jamia Millia University, New Delhi - 110025

NAME : YASH VINAYVANSHI

PROGRAM NAME : B.TECH (COMPUTER ENGINEERING)

SEMESTER : 6

EXAMINATION ROLL NO. : 19BCS081

UNIQUE PAPER CODE : CEN - 604

PAPER TITLE : EMBEDDED SYSTEM

NO. OF PAGES :

DATE OF EXAM : 04/06/2022

TIME OF EXAM : 10AM - 1PM

Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS081
FoE&T Semester: 6 Session: 2021-22
Jamia Millia University, New Delhi - 110025

A1 a (i) General Purpose Computing System (or Personal Computer)

1. A system which is a combination of a generic hardware and a general purpose OS for executing a variety of Applications.
2. Contains a General purpose operating Sys^m (GPOS)
3. Applications are Alterable (programmable) by the user (its possible for the end user to reinstall OS, end also add or remove users application)
4. Performance is the key deciding factor in the selection of system. Always Faster is Better
5. Less / Not at all tailored towards reduced operating power requirements.
6. Response requirements are not time critical.
7. Need not be deterministic in execution behaviour

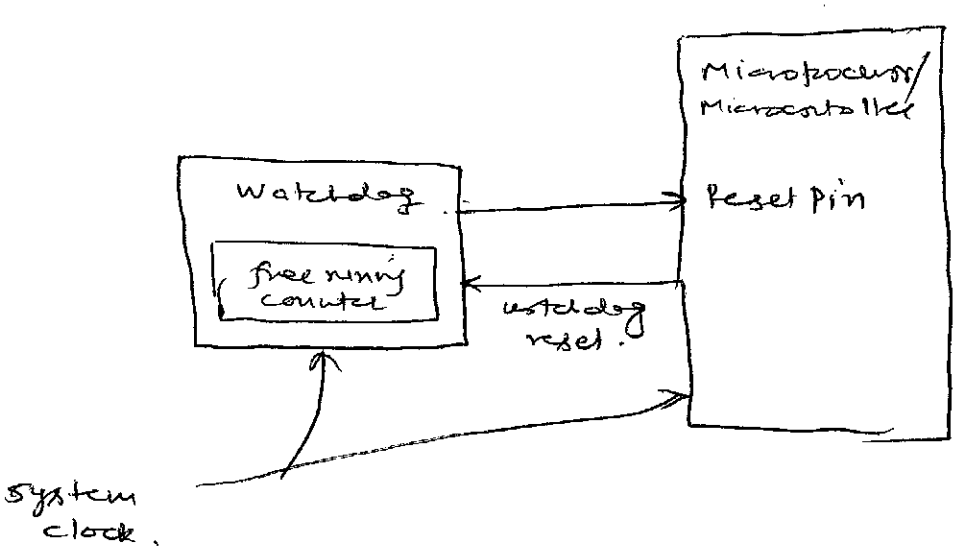
Embedded System.

1. A sys^m which is a combination of spl. purpose hardware & embedded OS for executing a specific set of Applications.
2. May or may not contain an OS for functioning
3. The firmware of the embedded system is pre-programmed and it is not alterable by the end user (There may be exceptions for sys^m supporting OS kernel image flashing through special hardware settings)
4. Appln specific requirements (like performance, power requirements, memory, usage etc) are the key deciding factors.
5. Highly tailored to take advantage of the power saving modes supported by the H/W & the OS.
6. For certain category of embedded systems like mission critical systems, the response time requirement is highly critical.
7. Exec^n behaviour can be deterministic for certain kinds of ES like hard real time system

Q1 a (ii) A watchdog timer is used to monitor the firmware execution and reset the system processor / microcontroller when the prm execution hangs up.

A watchdog timer increments or decrements a free running counter with each clock pulse and generates a reset signal to reset signal to reset the processor if the count reaches highest value or 0 value in up count or down count watchdog timer.

if watchdog counter is enabled, the firmware can write a 0 to it before starting the execn of a piece of code (which is susceptible to hang up) and watchdog will start counting, if firmware execn doesn't complete due to malfunctioning, within the time required by watchdog to reach maximum count, the watchdog will generate a reset pulse which'll reset the processor.

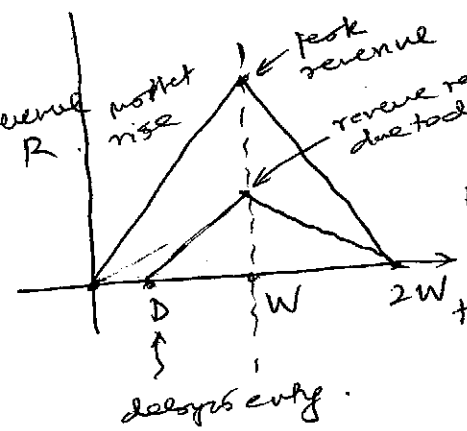


Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS 081
FoE&T Semester: 6 Session: 2021-22
Jamia Millia University, New Delhi - 110025

Q1 b)

Scenario 1 : General purpose processor.

NRE = ₹5000
unitcost = ₹30.
Time to market = 15 Months (120 - 15 = 105 months of market window remaining)



Revenue = $\frac{1}{2}$ × base × height
= $\frac{1}{2}$ × 120 × 125,000 ₹/month.
(120 months)
= ₹7500000

Fraction revenue loss
= (D * (3W - D)) / 2W²
= (15 × (3 × 60 - 15)) / (2 × 60²)
= $\frac{15 \times 165}{2 \times 3600}$ = 0.34375.

Total profit = $\left[\frac{[\text{revenue} \times (1 - \text{frac rev. loss})] - \text{NRE}}{\text{unitcost} + \phi}\right] \times 5$

ϕ being No cost added for profit.

= $\left(\frac{7500000 * (1 - 0.3438) - 5000}{30}\right) \times 5$

= $\left(\frac{4921875 - 5000}{30}\right) \times 5$.

= ₹ 8,19,479

Scenario 2 : Special purpose processor.

NRE = ₹ 20000
unitcost = ₹ 10
Time to market = ₹ 30 (120 - 30 = 90 months of market window remaining).

Revenue : Already calculated above.

Fraction revenue loss. = (D × (3W - D)) / 2W²
= (30 × (3 × 60 - 30)) / 2 × 60²
= $\frac{30 \times 150}{2 \times 3600}$ = 0.625

$$\text{Total profit} = \left(\frac{7500000 * (1 - 0.625) - 20000}{\text{unit cost} \rightarrow 10 + 0} \right) \times 5.$$

$$= \boxed{₹\ 13,96,250}$$

Although time to market is SP is more as well as NRE is high than GP, the unit cost compensate more and net profit in special purpose system is higher

Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS051
FoE&T Semester: 6 Session: 2021-22
Jamia Millia University, New Delhi - 110025

Q2a)

```
        ORG 0.
        MOV DPTR, #1000H
        MOV R0, #0AH.
        MOV R1, #20H
        MOV R2, #00H.
HERE:   MOV A, @DPTR.
        MOV B, #02H.
        DIV AB.
        MOV R3, B.
        CJNE R3, #0H, ODD.
        ACALL STOREEVEN.
        DJNZ R0, HERE.
ODD:    INC A
        ACALL STOREEVEN
        DJNZ R0, HERE
```

// counter for 10 nos
// R1 R2
// 20 00 H

// fetch to store even no.

// incrent A to make it even

```
STOREEVEN:  MOV R4, DPH.
            MOV R5, DPL
            MOV DPH, R1.
            MOV DPL, R2.
            MOVX @DPTR, A.
            INC DPTR.
            MOV R1, DPH
            MOV R2, DPL
            MOV DPH, R4
            MOV DPL, R5
            INC DPTR.
```

// store copy of DPTR in R4, R5
// put copy of DPTR in 2000+ series from R1 R2 into it
// store at position 2000.
// incrent DPTR
restore DPTR in 1000+ series.

→ stacks can also be used to store copies of DPTR and use it for extend memory in multiple places.

Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS 0 8 1
FoE&T Semester: 6 Session: 2021-22
Jamia Millia University, New Delhi - 110025

(Q 2b) 8051 can support only 64K bytes of external data memory since DPTR is 16 bit in size ($2^{16} = 64K$).

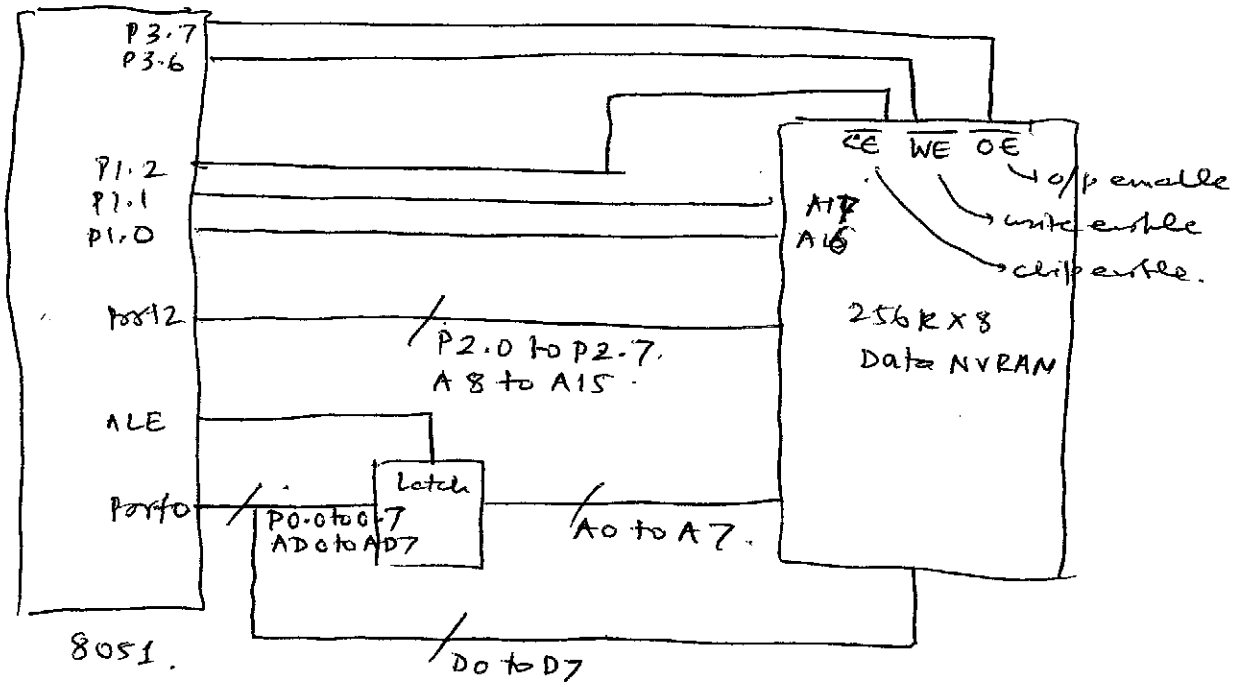
To connect more than 64K ie 256 k external memory, we connect A0 to A15 of 8051 directly to external memory's A0 – A15 pins and use some of the P1 pins to access the 64 k bytes blocks inside the single 256 K X 8 Memory chip

The 256 K X 8 NV-RAM has 18 address pins (A0 to A17) and 8 data lines. A0 to 15 go directly to mem chip. while A16 & A17 are controlled by P1.0 & P1.1 respectively. Also, the chip select of external Ram is connected to P1.2 of 8051

Chip select P1.2	A17 P1.1	A16 P1.0	Block addr space
0	0	0	00000H – 0FFFFH
0	0	1	10000 H – 1FFFFH
0	1	0	20000H – 2FFFFH.
0	1	1	30000H – 3FFFFH
0	X	X	external RAM disabled.
1.			

∴ To access the 20000 H – 2FFFF H address space we've to write following pgm.

```
CLR P1.2          // enable external RAM
MOV DPTR, #0.     // start of 64k mem block.
CLR P1.0          // A16 = 0
SETB P1.1         // A17 = 1 for 20000H block
MOV A, SBUF       // get data from serial port
MOV @DPTR, A      // save data in block 20000H address
INC DPTR.         // next loc
:
:
```


P3.7
P3.6
P1.2
P1.1
P1.0
port2
ALE
Port0
8051.
P2.0 to P2.7.
A 8 to A15.
Latch
P0.0 to 0.7
AD0 to AD7
A0 to A7.
D0 to D7
CE WE OE
o/p enable
write enable
chip enable.
A16
256K x 8
Data NVRAM

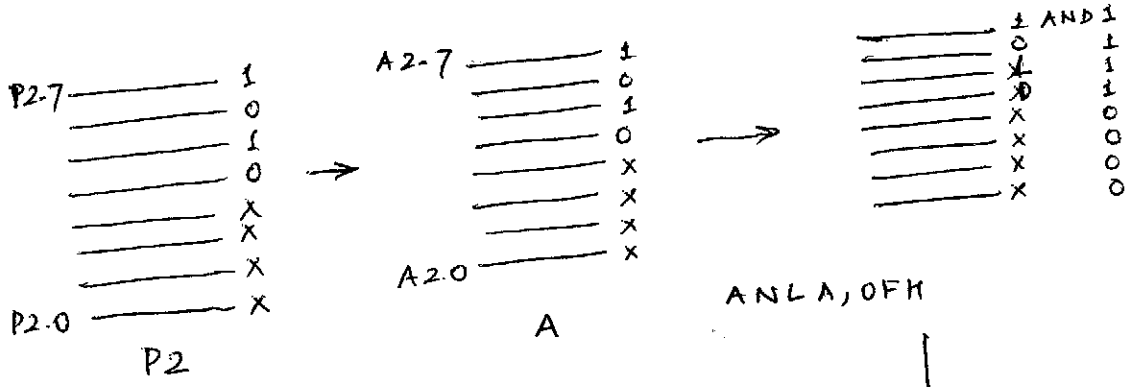
Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS 081
FoE&T Semester: 6 Session: 2021-22
Jamia Millia University, New Delhi - 110025

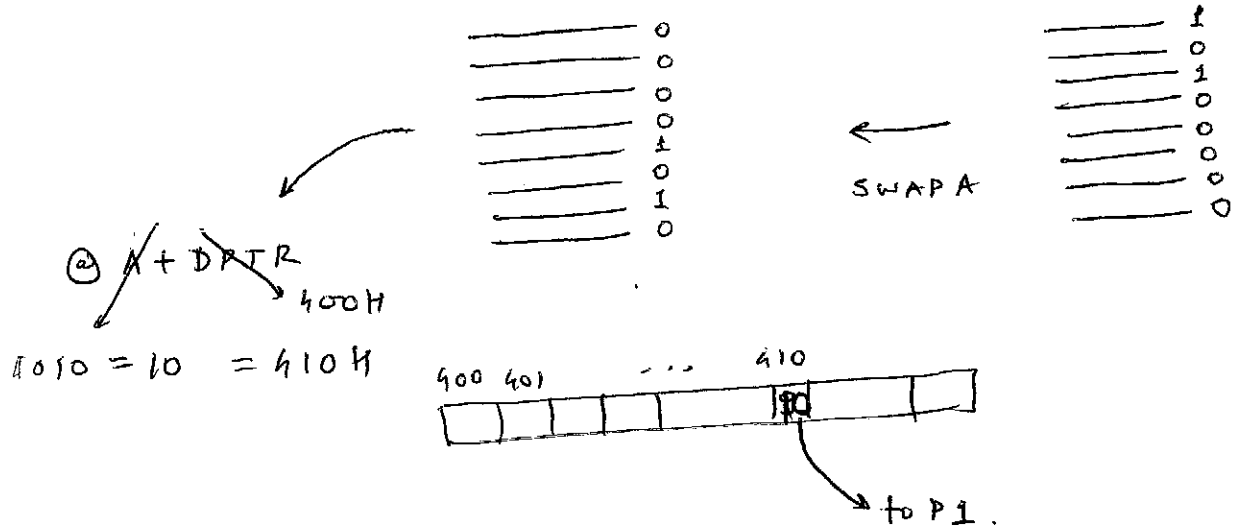
Q3 a) Given, upper 4 bits of P2 are connected to 4 switches

```
MOV DPTR, #CONVTAB.      // get SW status
MOV A, P2                // mask upper nibble.
ANL A, #0FH              // higher bits become lower bits
SWAP A                   // get data from lookup table
MOVC A, @A+DPTR.         // display value in P1
MOV P1, A                // stay here.
SJMP $.

        ORG 400H.
CONVTAB : DB 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.
        END.
```

Illustration





Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS081
FoE&T Semester: 6 Session: 2021-22
Jamia Millia University, New Delhi - 110025

Q3 b)

```
        ORG 0000.
        MOV DPTR, #200H.        // base address of data in ROM
        MOV R0, #64H.           // counter with (100)D.
        MOV R1, #00H.           // counter to keep count of
                                   nos within range.

HERE:   CLR A
        MOVC A, @A+DPTR.
        CJNE A, R2, NEXT.       // A ≠ R2.
        INC R1                  // A = R2 ⇝ inc
        JMP OUT

NEXT :  JNC OUT.                // A < R2.
        CJNE A, R3, NEXT1       // A > R2 & A ≠ R3.
        INC R1                  // A > R2 & A = R3.
        JMP OUT

NEXT1 : JC OUT                  // A > R3 & A > R3.
        INC R1                  // A > R2 & A < R3.
        JMP OUT.

OUT :   INC DPTR                // point to next address
        DJNZ R0, HERE           // until 100 bytes examined.
        MOV A, R1               // copy count into A.
        MOV P2, A.              // show count on P2
```

The program increments count when

lower limit (content of R2) $\leq$ current byte $\leq$ upper limit (content of R3)

Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS081
FoE&T Semester: 6 Session: 2021-22
Jamia Millia University, New Delhi - 110025

Q4a) Timer's clock period

$f = 1/12 \times 22\,MHz \approx 1.833\,MHz$

$T = 1/1.833\,MHz \approx 0.546\,\mu s$.

HCF(40 ms, 30 ms, 15 ms) = 5 ms.

Timer initializ^n for 5 ms delay.

$$\frac{5\,ms}{0.546\,\mu s} \approx 9158$$

$65536 - 9158 = 56378 =$ **DC3A H**

Approach:
- To generate 40 ms delay we've to generate 5 ms delay 8 times
- similarly for 30 & 15 ms) we've to generate 5 ms delay 6 & 3 times respectively.
- we write a function which generates 5 ms delay the no. of times as stored in R0.

```
        MOV TMOD, #01H      // timer 0 Mode 1 (16 bit).
        CLR P1.5            // waveform starts from low
HERE:   MOV R0, #08H        // init R0 to 8
        ACALL DELAY         on // run 5 ms delay for 8 times thru DELA
        MOV R0, #08H        // init R0 to 8
        ACALL DELAY         off // run 5 ms delay for 8 times
        MOV R0, #06H        on // init R0 to 6
        ACALL DELAY         // run 5 ms delay for 6 times
        MOV R0, #06H        off //  "
        ACALL DELAY         //  "
HERE1:  MOV R1, #04H        // initialise R1 to 4 as 4 pulses to gen
        MOV R0, #03H        // init R0 to 3
        ACALL DELAY         // run 5 ms for 3 times => 15 ms dela
        MOV R0, #03H        on //  "
        ACALL DELAY         off //  "
        DJNZ R1, HERE1      // repeat until counter of 4 exhausts
        JMP HERE            // continue next process again

DELAY:  CPL P1.5            // invert P1.5
HERE2:  MOV TL0, #3AH       // initialise timer 0 for 5 ms delay
        MOV TH0, #DCH
        SETB TR0            // start timer 0
AGAIN:  JNB TF0, AGAIN      // wait until timer 0 exhausts
        CLR TR0             // stop timer 0
        CLR TF0             // clear TF flag
        DJNZ R0, HERE2      (Next line: RET) // do it R0 no. of times, retu
```

Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS081
FoE&T Semester: 6 Session: 2021-22
Jamia Millia University, New Delhi - 110025

Q5 a) (i) XTAL = 22MHz

∴ Timer's clk period = 0.546 μs

To use: timer 1, mode 1 (16 bit).

No of clock cycles for 0.25 sec exceed 65536.

250 ms.

Approach is to generate 25 ms delays 10 times.

Timer initialization for 25 ms.

$$\frac{25\text{ ms}}{0.546\ \mu\text{s}} = 45787$$

65536 − 45787 = 19749 = (4D25)H

(ii) To generate boudrate of 4800 b/s

We use timer 0 in mode 2.

$$\frac{22\text{MHz}}{12 \times 32} = 57292\text{Hz} \approx 57600\text{Hz}$$

$\frac{57600}{9600} = 6$

↑ −6 or FAH

(iii) P3.2 ≡ INT 0
P3.3 ≡ INT 1

```
        ORG 0000H
        LJMP MAIN
                                // jump to ISR to blink led P3.0.
        ORG 0003H
        LJMP LED1
                                // jump to ISR to blink led P3.1.
        ORG 000BH
        LJMP LED2
                                // jump to ISR to complement or toggle
        ORG 001BH                  bit P3.1 & reload timer1 of 0.25
        LJMP WAVE
                                // jump to ISR for serial com.
        ORG 0023H
        LJMP SERIAL

        ORG 0030H.
MAIN:   MOV P2, #0FFH           // Make P2 an input port.
        MOV TMOD, #00010010     // timer1 mode 1 timer0 mode 2
                                // boudrate = 9800 b/s
                                ] Init timer 1 for generating 25 ms
                                  delay.
        MOV TH0, #0FAH          // to run 25 ms delay 10 times
        MOV TH1, #4DH
        MOV TL1, #25H
        MOV R3, #0AH
        MOV SCON, #50H.         // 8 bit, 1 stop, REN enabled.
        MOV IE, 10011101        // INT0, INT1, T1, RI/TI interrupts
        SETB TR0.               // start timer 0
        SETB TR1                // start timer 1.
HERE:   MOV A, P2               // transfer input port from P2 to A
        MOV P1, A.              // give a copy to P1
        MOV SBUF, A             // give a copy to SBUF to start serial
        SJMP HERE                  sending
                                // continuously.
```

```
        ORG 100.
LED1:   SETB P3.0
        MOV R0, #255.
BACK:   DJNZ R0, BACK
        CLR P3.0.
        RETI.

LED2:   SETB P3.1
        MOV R1, #255.
BACK:   DJNZ R0, BACK
        CLR P3.1
        RETI

SERIAL: JB TI NEXT
        MOV A, SBUF
        CLR RI.
        RETI.
NEXT:   CLR TI.
        RETI.

WAVE:   ~~MOV A,R3~~
        JNZ R3, HERE2
        CPL P2.1.
        MOV R3, #DAH
        ~~DJNZ R3~~

HERE2:  MOV TH1, #4DH
        MOV TH1, #25H.
        CLR TR1
        CLR TF1.
        DECR3
        SETB TR1
        RETI.

        END.
```

// turn LED at P3.0 on
// wait for 255 x 0.546 μs

// turn off LED at P3.0.
// return

Similar for LED at Pin P3.1

// if transfer complete
// otherwise due to receive
// clear RI since CPU does not return for ISR
// if transfer complete, clear TI flag
// return & enable interrupts of equal or low priorities than this interrupt

// check if R3 is exhausted ie timed for 25 ms ran 10 times
// if yes, toggle pin 2.1 as delay of 250 ms accumulate
// restore R3 with value 10.

] initialise timer 1 for 25 ms delay

// stop timer 1
// clear TF FLAG.
// Timer ran once more so decrease R3
// start timer 1 again
// return

// ok.

Yash Vinayvanshi
B. Tech. (Computer Engineering)
Roll No.: 19BCS081
FoE&T Semester: 6 Session: 2021-22
Jamia Millia University, New Delhi - 110025

Q5b)

```
MOV TMOD, #20H.        // timer 1 mode 2.
MOV TH1, -6            // 4800 bauds rate XTAL = 11.059MHz
MOV SCON, #50H.        // 8 bit, 1 stop, REN enabled
MOV R0, #40H           // R0 is ptr to internal RAM
MOV R1, #0AH.          // R1 is counter of bytes.
MOV A, #0C4H.
SETB A.0.
MOV #0C4H, A  : // Enable access to 1KB SRAM.
MOV DPTR, #200H.       // DPTR is ptr to SRAM
SETB TR1.              // start timer 1.
HERE: JNB RI, HERE     // wait for byte to come in
      MOV A, SBUF.     // store byte in A
      MOV B, A.        // store copy in A.
      RLC A            // Rotate left with copy A
      JC NEG           // if MSB = 1 ~ NO. is -ive
      MOV A, B.        // if no. positive restore it to A
      MOV @R0, A       // move it to RAM
      INC R0.          // increment R0
      JMP NEXT         // continue.

NEG : MOV A, B         // if no. is -ive, restore it in A
      MOVX @DPTR, A    // store it in SRAM
      INC DPTR.        // increment DPTR

NEXT: CLR RI.          // get ready to receive next byte
      DJNZ R1, HERE.   // do it 10 times.
```