QUESTION 1) [10 points]

```
STRING RMB 30
  DAT $54, $68, $69, $73, $20, $69, $73, $20
  DAT $61, $20, $74, $65, $78, $74, $00
*Example string="This is a text0"

LETTER RMB 1
  DAT $74
* Example searched letter="t"

COUNT RMB 1

START
  LDA SK, STRING      ;Get address of string
  STA 0, COUNT        ;Initialize count of founds
LOOP
  LDA A, <SK+0>   ;Get a letter from String array
  INC SK          ;Increment SK
  CMP A, 0        ;Compare to 0 (End of string?)
  BEQ END
  CMP A, <LETTER> ;Is it equal to searched letter?
  BNE LOOP        ;Continue if not equal
  INC <COUNT>     ;Increment count of founds
  BRA LOOP
END INT
```

QUESTION 2) [30 points]

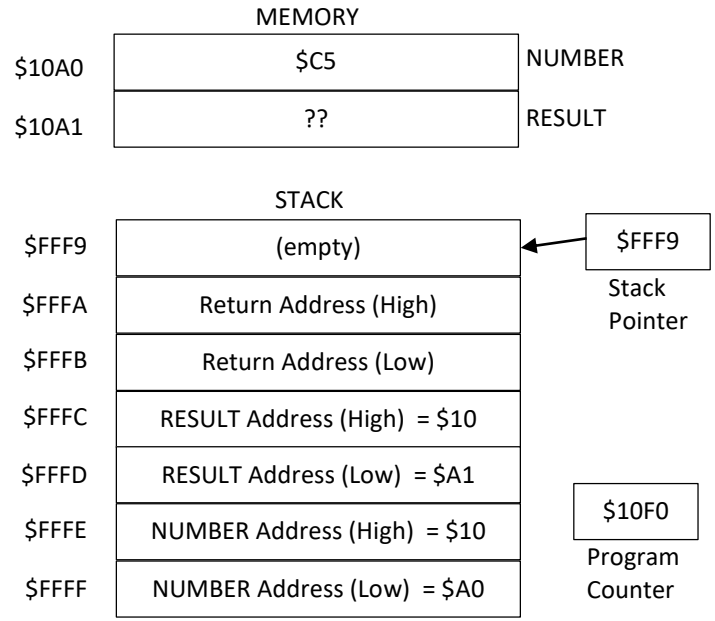
a) [10 points]

```
NUMBER RMB 1
  ORG $10A0
  DAT $C5     ;(1100 0101)
RESULT RMB 1    ;Will be $A3 (1010 0011)

*Main program
  ORG $10B0
START
  LDA YG, $FFFF

*Pass the addresses of variables via stack
*(Call-by-reference method)

  LDA AB,NUMBER ;AB now contains an address
  PSH B         ;Low byte of NUMBER address
  PSH A         ;High byte of NUMBER address

  LDA AB,RESULT  ;AB now contains an address
  PSH B          ;Low byte of RESULT address
  PSH A          ;High byte of RESULT address

  BSR REVERSE_THE_BITS      ;Call subroutine
  PUL A
  PUL A
  PUL A
  PUL A
  INT
```

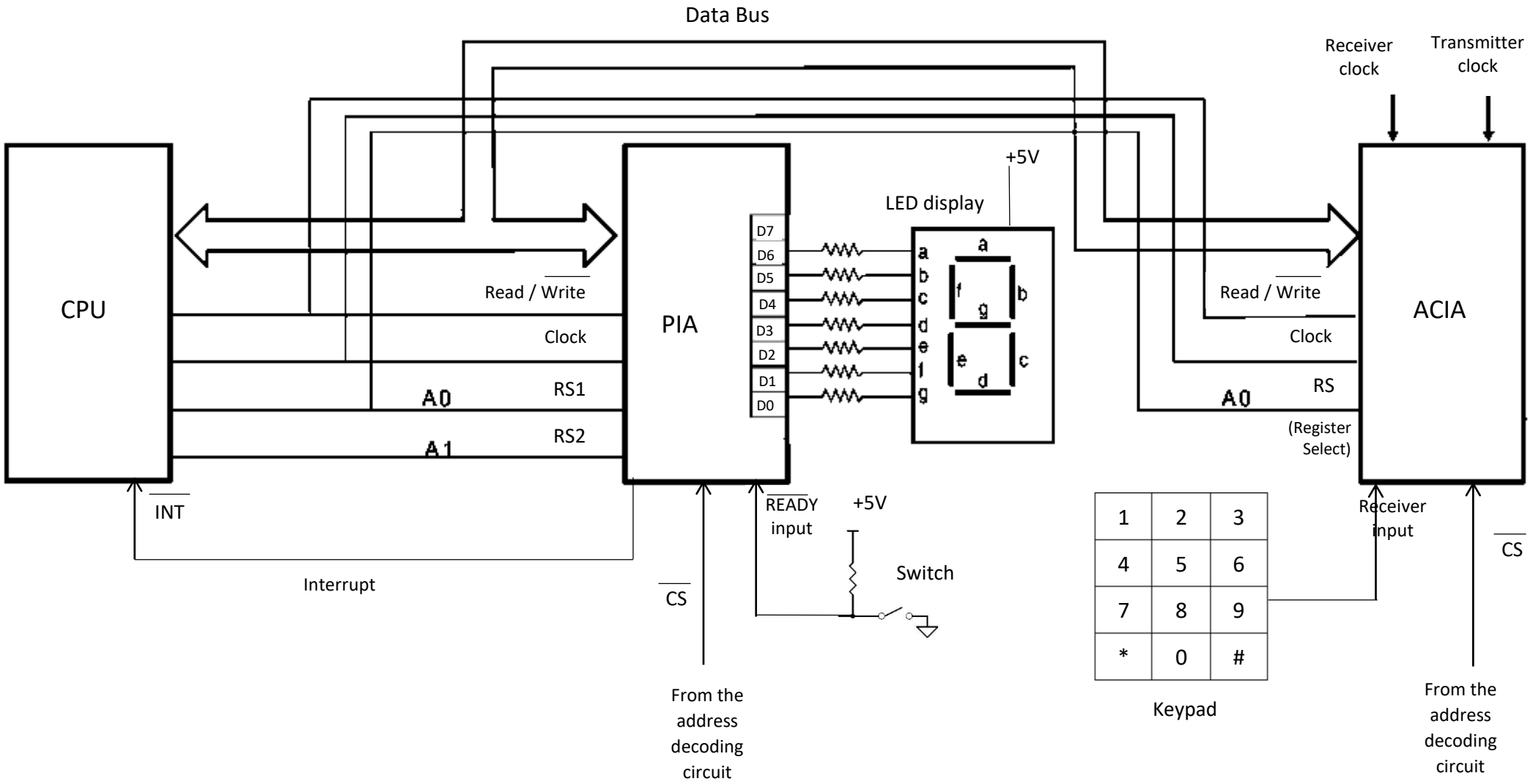
b) [10 points]

```
  ORG $10F0
REVERSE_THE_BITS
*Read the passed addresses of variables from stack.
  LDA SK,<YG+05>  ;Get NUMBER address on stack
  LDA A, <SK+0>     ;Read the data itself
  LDA  C, 0         ;Count until 8

LOOP
  LSL A    ;Logical shift left (MSB of A goes to Carry flag)
  ROR B   ;Rotate right (Carry flag goes to MSB of B)
*          B is used to obtain the reversed bits.
  INC  C   ;Increment the C counter
  CMP C, 8  ; Compare counter to 8
  BLT  LOOP ; Continue if C-8 is less than 0

  LDA SK,<YG+03>    ;Get the RESULT address on stack
  STA B,<SK+0>      ;Save reversed bits to the RESULT
  RTS               ;Return from subroutine
```

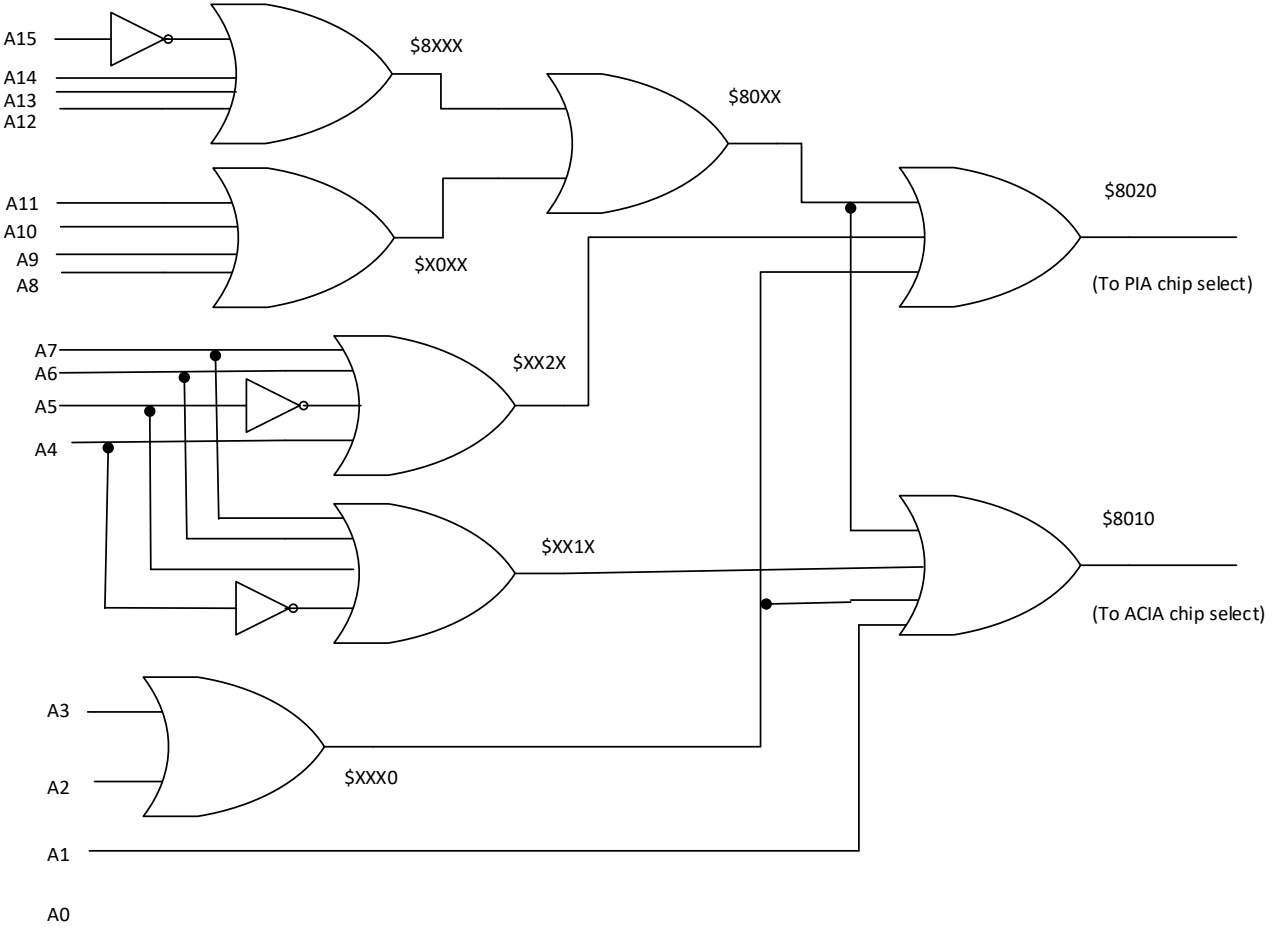
c) [10 points]



QUESTION 3) [60 points]
a) [20 points]



b) [10 points] Address Decoding Circuit



c) [30 points]

```
ACIA_STATCON    EQU     $8010
ACIA_RX_TX      EQU     $8011
PIA_PORT        EQU     $8020
PIA_DIRECTION   EQU     $8021
PIA_STATCON     EQU     $8022
*-------------------------------------------------
*Character table for 7-Segment LED Display.
*LED type : Common anode (common voltage),
*A segment lights when logical 0.
*(a-g --> D6-D0)  D7 is not used.
LED_CHAR_TABLE RMB 10
  DAT %1000 0001        ;0
  DAT %1111 1001        ;1
  DAT %1010 0100        ;2
  DAT %1011 0000        ;3
  DAT %1001 1100        ;4
  DAT %1001 0010        ;5
  DAT %1000 0010        ;6
  DAT %1111 1000        ;7
  DAT %1000 0000        ;8
  DAT %1001 0000        ;9
*-------------------------------------------------
*Main program
START
*Set the interrupt vector with Interrupt
*Service Routine address.
        STA SWITCH_ISR, <$FFF0>
        EIN    ;Enable interrupt

        BSR CONDITIONING_ACIA
        BSR CONDITIONING_PIA

READ_KEYPAD
        BSR READ_ACIA   ;A contains non_ASCII key
```

```
* (Continued main program)
DOWN_COUNTING_LOOP
        LDA C, $0
        MOV D, A ;A is used as counter.
        LDA SK, LED_CHAR_TABLE
        LDA B, <SK+CD+0>
*Get corresponding LED character from Table

        STA B, PIA_PORT
*Write LED character to port register in PIA
    BSR WAIT
    DEC A       ;Counting down
    BNE DOWN_COUNTING_LOOP

    BRA READ_KEYPAD
*End of main program
```

```
CONDITIONING_ACIA
*T1 T0 =  0 1 (Speed:1/2)
*T3 T2 =  1 1 (Data + Stop = 8+2 bits)
*T5 T4 =  0 1 (Parity: Odd)
*T7 T6 =  0 0 (Not used here)
    STA %00011101, ACIA_STATCON
    RTS
```

```
CONDITIONING_PIA
    STA $FF, PIA_DIRECTION
*All bits of PIA are for output (LED)

*D1 D0 =  1 0
*When READY input of PIA goes from 1 to 0,
* PIA will generate interrupt.
    STA $02, PIA_STATCON
    RTS
```

```
READ_ACIA
        LDA B, <ACIA_STATCON>

        AND B, $01 ; D0 bit taken
*Is there any data in RX register of ACIA?
        BEQ READ_ACIA  ;Status not ready yet

        LDA A, <ACIA_RX_TX>
*Read data from receiver register in ACIA

        AND A, $0F
*Convert Keypad value from ASCII to non-ASCII
        RTS
```

```
*Interrupt Service Routine for Switch
SWITCH_ISR
    LDA A, $0
*This will stop the down counting
        RTI
```

```
* WAIT Routine
WAIT
    LDA SK,20000
LOOP  DEC SK
    BNE LOOP
    RTS
```