Memo

Question 1:

Need to calculate the conversion from the LM35 to a byte. We have Vrefh = 2.56V and Vrefl = gnd so the least bit is:

dV = 2.56/256 = 0.01V

i.e. one bit = 1 °C.

Rough flowchart on next page.

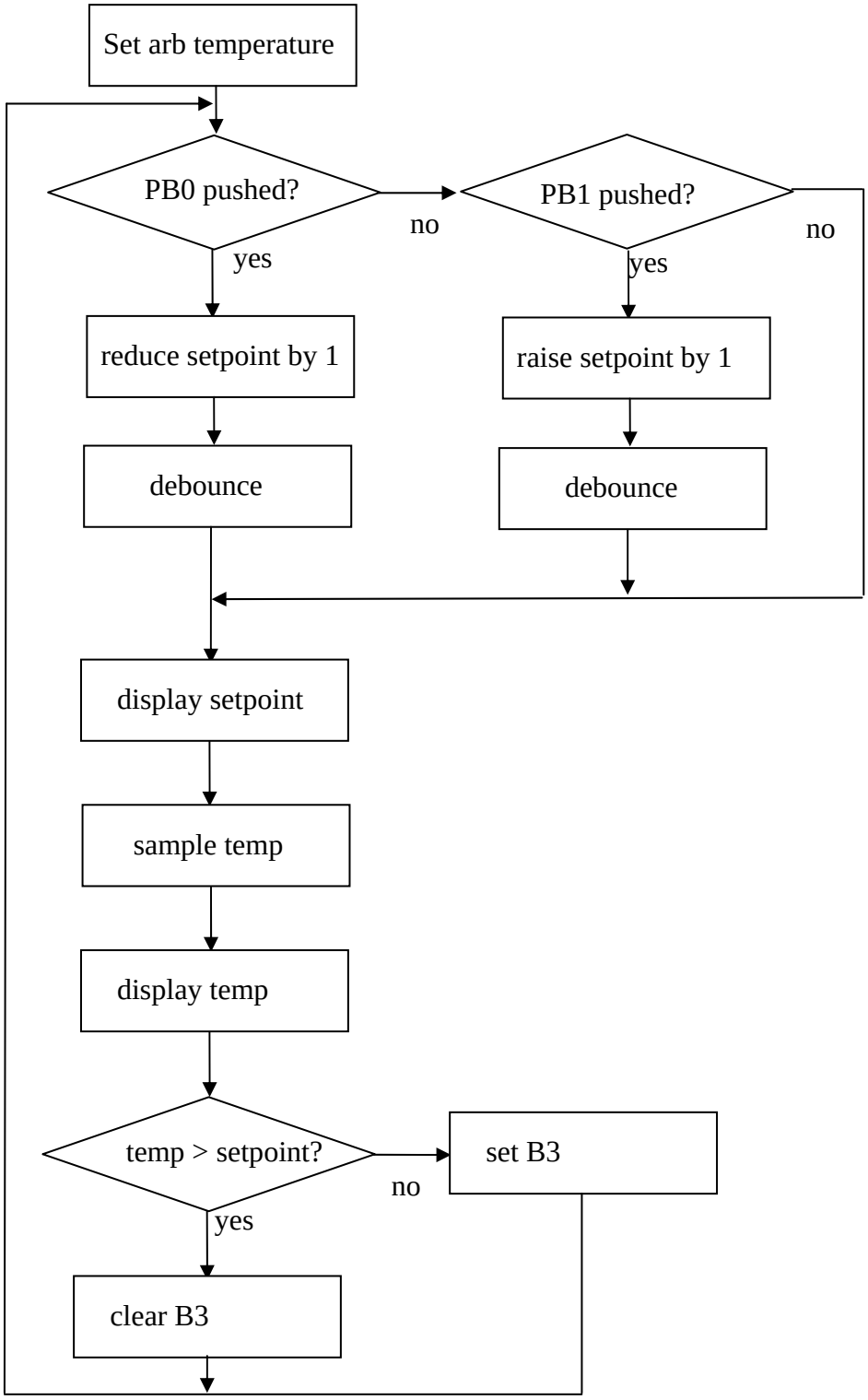
Code:

```
        mov     #$08,DDRB           ; all Port B inputs except B3
        mov     #$FF,DDRD           ; all port D outputs
        jsr     Init_LCD            ; initialize LCD
        jsr     Clear_LCD           ; clear LCD

        mov     #$14,count          ; use count as setpoint variable, set to 20 degrees
next:   brclr   0,PTB,down_one      ; if PB0 pushed reduce setpoint
        brset   1,PTB,no_change     ; if PB1 not pushed, don't change
        jsr     Delay_1sec          ; PB1 pushed so debounce with delay
        inc     count               ; and raise setpoint
down_one:
        jsr     Delay_1sec          ; use delay as debounce
        dec     count               ; reduce setpoint by one
no_change:
        jsr     out_set             ; display setpoint
        jsr     GetADC_B2           ; sample temp
        jsr     out_temp            ; display temperature
        cmp     count               ; compare temp (in Acc) with setpoint
        bge     turn_off            ; if greater or equal, turn heater off
        bset    3,PTB               ; or turn on
        jmp     next                ; and start again
turn_off:
        bclr    3,PTB               ; turn off
        jmp     next                ; and start again

out_set:
        lda     ??
        jsr     Write_LCD1  ( put some ascii character here, e.g spell out "setpoint=")
        etc…
        etc…
        lda     count
        jsr     Write_LCD1
        rts
```

same for out_temp.


Set arb temperature
PB0 pushed?
no
PB1 pushed?
no
yes
yes
reduce setpoint by 1
raise setpoint by 1
debounce
debounce
display setpoint
sample temp
display temp
temp > setpoint?
no
set B3
yes
clear B3

Question 2:

a) 3 bits
b) 20/8 = 2.5V
c) Electronic conversion is effectively instantaneous; time is limited by the logical decoding in the microprocessor.
d) Ignore B7 and A3.

B6	B5	B4	B3	B2	B1	B0	A2	A1	A0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	1	0	1	0
0	0	0	0	1	1	1	0	1	1
0	0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1

e) Vin = 6.3 V
   - ⇨ (10+6.3)/2.5 = 6.5
   - ⇨ output will be 110

f) Many different ways to do this. One possibility

```
        mov   #$00,DDRB        ; Port B all inputs
        mov   #$FF,DDRA        ; Port A all outputs

        lda   PTB              ; get inputs
        and   #$7F             ; set B7 to zero
        sta   Timeout1         ; store input state
        clr   Count            ;set count to zero

next:   asr   Timeout1         ; shift right, least bit goes into carry
        bcc   finished         ; if carry zero, we're done
        inc   Count            ; else increment output
        jmp   next             ; and do again

finished:
        mov   Count,PTA        ; Count is output
```

g) maximum case will take 8 x asr, bcc instructions:

Instruction	Occurrences	Cycles	Total
mov	3	4,4,5	13
lda	1	3	3
and	1	2	2
sta	1	3	3
clr	1	5	5
asr	8	5	40
bcc	8	3	24
inc	7	5	35
jmp	7	3	21
		Total	146

Question 3:
a)

```
delay4:                                         del1  del2
       mov #$03,del1          4                 3
       bra delay2             4
delay5:
       mov #$FF,del1
       bra delay2
delay1:
        mov #$A0,del1
delay2: mov #$03,del2         4                       3
delay3: nop                   1,1,1
        nop                   1,1,1
        nop                   1,1,1
        dbnz del2,delay3      5,5,5                   2,1,0
        dbnz del1,delay2      5                 2
        rts
```

Inner loop takes 8x3 cycles. Outer loop takes 9 cycles => 33 cycles, runs 3 times => 99 cycles + 6 for jsr, 8 for first 2 lines, 4 for rts => 99+16 = 115 cycles.

b) inner loop above takes 8 cycles => run 50 times; but need some extras, so run 47 times (8x46= 368) and then check for 32 extra cycles:

```
        jsr delay4            ;6
delay4:
       mov #$03,del1          ;4
       bra delay2       ;4
delay2: mov #$2D,del2         ;4     $2D = 46
delay3: nop                   ; do 46 times =46
        nop                   ; do 46 times =46
        nop                   ; do 46 times =46
        dbnz del2,delay3      ; do 46 times = 230+46x3 =368
        dbnz del1,delay2      ;5
        rol dummy             ;3
        mov #$00,dummy        ;4
        rts                   ;6  = 400 cycles
```

c) Hexadecimal 6E A0 80
d) Hexadecimal 20 03 (skips the next three bytes).