100

You must show how you get your answer for a full credit in each problem. The final answer only will not get a full credit even if correct.

1. Represent the following **decimal** integers in the signed 2's complement binary number system which uses **9 bits** for each number. And then express them in **12 bits** through sign-extension.
   - 133
   - -200

decimal	binary (9-bit)	binary (12-bit)
133	010000101	000010000101
-200	100111000	111100111000

$2^9 - 200 = 512 - 200 = 312$

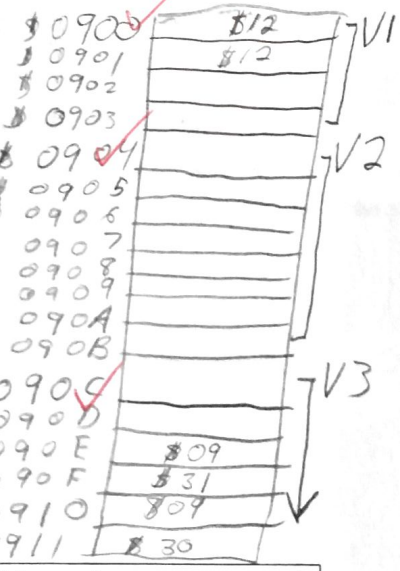
133: 133/2 = 66 R1, 66/2 = 33 R0, 33/2 = 16 R1, 16/2 = 8 R0, 8/2 = 4 R0, 4/2 = 2 R0, 2/2 = 1 R0, 1/2 = 0 R1 → 010000101

312: 312/2 = 156 R0, 156/2 = 78 R0, 78/2 = 39 R0, 39/2 = 19 R1, 19/2 = 9 R1, 9/2 = 4 R1, 4/2 = 2 R0, 2/2 = 1 R0, 1/2 = 0 R1 → 100111000

2. (a) Draw the memory map where the variables, V1, V2 and V3 with their starting addresses are shown.
   (b) For each of the 6 highlighted instructions, derive the effective address (in hexadecimal) of the memory operand (source or destination) involved. Assume that all variables are initialized to zero.

```
V1      ORG   $900
        DS.B  4
V2      DS.W  4
V3      DS.B  100

        ORG   $4000
        movb  #$12,V1
        ldx   #V1
        ldaa  0,x
        staa  1,x
        ldy   #$0930
        ldab  #4
        ldx   #V3
        sty   b,x
        iny
        sty   2,+x
        ldaa  2,x-
        ldd   #2
        ldaa  [d,x]
```



instruction	effective address of memory operand
movb #$12,V1	$12 → $0900
ldaa 0,x	$0900
staa 1,x	$12 → $0901
sty b,x	#$0930 → $0910
sty 2,+x	#$0931 → $090E
ldaa [d,x]	$0931

#$0900 → x
$12 → a
$12 → $0901
#$0930 → y
#04 → b
#$090C → x
#$0931 → y
#$090E → x
#$0931 → $090E
#$09 → a
#$090C → x
#$0002 → d

1,x
↳ x+d = #$090E

EA: $0931 ($0931) = $00 → a

3. For each of the arithmetic instructions, derive the result of arithmetic operation and determine if each flag is set ("1") or cleared ("0").

```
ldaa   #$40
adda   #$50
ldab   #$70
addb   #$D0
ldaa   #$A0
suba   #$30
ldab   #$30
subb   #$70
```

$7F | 127

$80 | −128

instruction	result of operation	Z	N	C	V
adda #$50	$90 → a	0	1	0	1
addb #$D0	$40 → b	0	0	1	0
suba #$30	$70 → a	0	0	0	1
subb #$70	$C0 → b	0	1	1	0

(+) $40 → a
(+) + $50
(−) $90 → a

0100

1001

(+) $70 → b
(−) + $D0
$140 → b

(−) $A0 → a
(+) − $30
$70 → a

$A0 → % 1010 . . .

borrow

(+) $30 → b
(+) − $70
$C0 → b

$C0 → % 1100 . . .

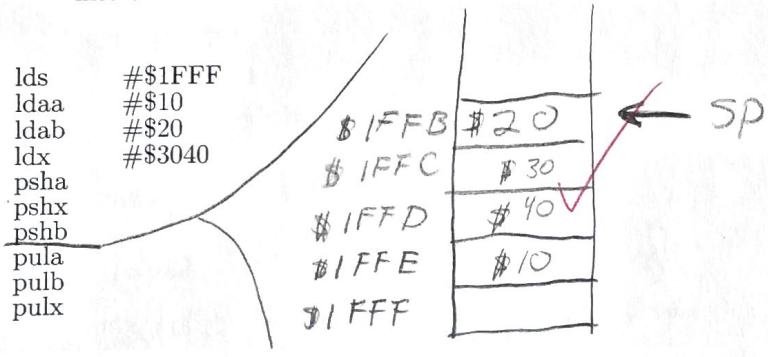
4. The stack pointer **sp** has been initialized to $1FFF.

   (a) Draw the stack right after the instruction **pshb** is executed, showing the top 4 bytes and their addresses.

   (b) What are the contents of the registers **a**, **b** and **x**, and the stack pointer **sp** right after each of the pull instructions?

$10 → a
$20 → b
$3040 → X

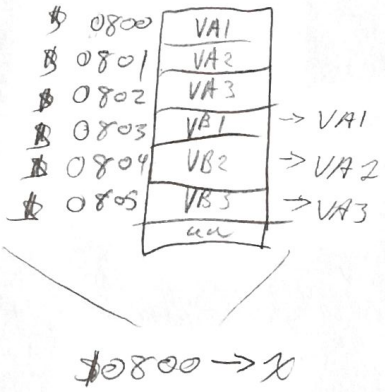
```
lds    #$1FFF
ldaa   #$10
ldab   #$20
ldx    #$3040
psha
pshx
pshb
pula
pulb
pulx
```



instruction	reg. **a**	reg. **b**	reg. **x**	stack pointer **sp**
pula	$20	$20	$3040	$1FFC
pulb	$20	$30	$3040	$1FFD
pulx	$20	$30	$4010	$1FFF

5. The array `VA` has been loaded with three 1-byte numbers. Write a program which copies the three numbers in `VA` into the respective locations of the array `VB`, **one byte at a time. You must use the constant-offset (indexed) addressing mode in copying.** Minimize the number of instructions.

```
VA      DS.B  3
VB      DS.B  3
        ORG   ROMSTART
        ldx   #VA
        movb  0,x  3,x
        movb  1,x  4,x
        movb  2,x  5,x
        rts
```



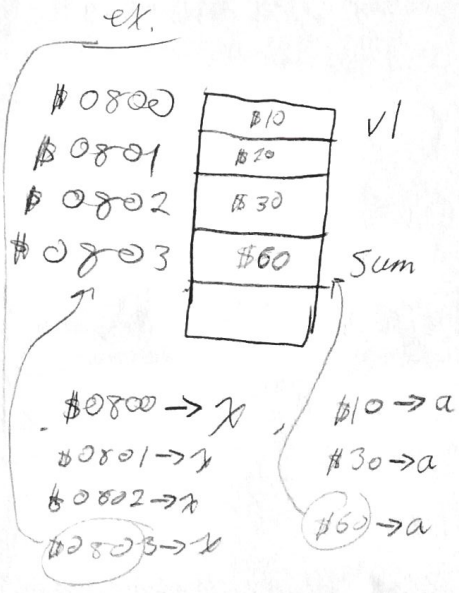
n, x
constant
↓
no increment/decrement

6. Write a program which computes the total sum of the three 1-byte numbers in `v1` and stores it in `sum` assuming no overflow. **You must use the auto-increment/decrement (indexed) addressing mode in accessing the numbers in `v1`.** Minimize the number of instructions.

```
v1      DS.B  3
sum     DS.B  1
        ORG   ROMSTART
        ldx   #v1
        ldaa  1,x+
        adda  1,x+
        adda  1,x+
        staa  0,x
        rts
```

num1 → a
x+1 → x
a + num2 → a
x+1 → x
a + num3 → a
x+1 → x

ex.



In this exam., the microcontroller HCS12 is assumed unless specified otherwise. You must show how you get your answer for a full credit in each problem. **The final answer only will not get a full credit even if correct.**

87

1. Write a program which copies the number in `N` into `ODD` only if it is an odd number, or into `EVEN` only otherwise (if it is an even number).

```
N       DS.B  1
ODD     DS.B  1
EVEN    DS.B  1
```

N is ODD if LSB = 1

```
        BRSET N, %00000001, isOdd
        MOVB  N, EVEN
        BRA   done
isOdd:  MOVB  N, ODD
done:   RTS
```

2. The array `samples` holds 100 unsigned 1-byte numbers. Write a program which adds all of the numbers in `samples` and stores the total sum in `TOTAL`. The 1-word total sum stored in `TOTAL` must be a valid result. Minimize the number of instructions.

```
samples DS.B  100
TOTAL   DS.W  1
```

```
        ldaa  #0          ; counter
        ldx   #samples
        ldy   #$0
loop:   addy  a,x         ; add to Y element in samples
        inca              ; increment offset
        cmpa  #100        ; see if a = 100 (decimal)
        bne   loop        ; if a ≠ 100 continue loop
        sty   TOTAL       ; store y into mem
```

3

3. The variable `TEST` holds a one-byte number which is one of the 10 different numbers (patterns) in the array `PATTERNS`. Write a program which consists of a main program and a subroutine `find_index`. The main program passes the address and size of the array `PATTERNS` to the subroutine through the stack. The subroutine finds the index of the pattern matching the number in `TEST` and returns it through the stack. The subroutine can access `TEST` directly. The main program stores the index in the variable `INDEX`. The indices of the first through the last patterns are 0 through 9, respectively. Minimize the number of instructions.

```
TEST      DS.B  1
PATTERNS  DS.B  10
INDEX     DS.B  1

        lds   #$2000  ;init SP
        ldx   #PATTERNS
        pshx          ;push X onto stack
        ldab  #$0A    ;size of PATTERNS
        pshb
        jsr   find_index
        pula
        staa  INDEX
        rts

find_index:
        pula          ;load return address
        pulb          ;load size of array
        pulx          ;load location of array
        psha
        tfr   x,y     ;y holds same as x
        ldaa  TEST
loop:   cmpa  0,x
        beq   found
        inx
        bra   loop
found:  subx  y
        tfr   x,a
        pula
        psha
        pshy
        rts
```

4. An input device generates bytes to be input through the port A. The control signal `NEW` from the device goes high ("1") when the device has a new byte to be input and is connected the MSB of the port B. Once the new byte is input, the control signal `NEW` is automatically lowered (to "0"). Write a program which reads in two new bytes from the device and stores them in `in_data`. Minimize the number of instructions.

```
PORTA     EQU   0
DDRA      EQU   2
PORTB     EQU   1
DDRB      EQU   3

in_data   DS.B  2
        ldx   #in_data
        ldy   #$0002
        ldab  #$00
spin:   brclr PORTB,%10000000,1,spin
        ldaa  PORTA
        staa  b,x
        inc b
        dbne  y,spin
        rts
```

5. There is a keyboard which generates the level-6 interrupt whenever a key on the keyboard is pressed. The interrupt request (signal) from the keyboard is connected to the IRQ pin and the ASCII code of a key is input through the port B. Write a program which inputs the ASCII codes of the first 10 keys pressed and stores them in the array KEYS. The interrupt service routine reads in each code and the main program stores it in the array.

```
PORTB   EQU   $0001
INTCR   EQU   $001E

        ORG   $800
Buffer  DS.B  1
KEYS    DS.B  10

        ORG   $2000
lds     #$4000
bset    INTCR, %0100 0010   ; IRQEN...?
cli
ldy     #$000A
ldx     #KEYS
ldab    #$00
loop:   wai
        ldaa    Buffer
        staa    b,x
        inc b
        dbne    y,loop
        rts

key_isr:
        ldaa    PORTB
        staa    Buffer
        rti
        ...
; interrupt vectors
        ORG     $FFF2
        DC.W    key_isr
        ...
```

6. Write a program which generates a delay of 2 seconds (use polling). The frequency of the clock applied to the timer counter (TCNT) is $2^{16} \times 100$ Hz.

don't know what each bit does...

```
TCNT    EQU   $0044
TSCR1   EQU   $0046
TSCR2   EQU   $004D
TFLAG1  EQU   $004E
TFLAG2  EQU   $004F

bset    TSCR1, $80  ; enable Timer
ldy     #200        ; counter
movb    #$80, TFLAG1 ; clear TOF
spin:   brclr   TFLAG1, %0000 0010, spin ; wait for TOF set
        movb    #$80, TFLAG1 ; clear TOF
        dbne    y, spin      ; decrement count and repeat
        rts
```

$2^{16} \times 100$ Hz $\Rightarrow T = \frac{1}{2^{16} \times 100}$ s

1 TOF = ?

$d = 2$ sec, $D = \frac{2}{\frac{1}{2^{16} \times 100}} = 2^{16} \times 200$ clocks or 200 TOF

13