Problem 9.1:

Write a subroutine *cnt_one* to determine the number of ‘1’s in a 16-bit word. A pointer to the 16-bit word to be tested is passed in register X. The result should be returned in accumulator A. The subroutine must not modify the word being tested and must save any registers or accumulators used by the subroutine (except for A). *(Hint: create an 8-bit local variable on the stack for counting the number of ones)*

Problem 9.2:

(a) The program below is executed from the address labeled *start*. Draw the stack frame showing the <u>position of the stack pointer</u> and the <u>address and contents</u> of each byte on the stack (where known) when the processor is executing the instruction labeled *test2*. (You do not need to determine the value of the return address(s). Simply indicate their position on the stack).

(b) Then redraw the stack frame showing the position of the stack pointer when the processor is executing the instruction labeled *test1*.

```
start:          lds     #$5400
                ldd     #$8A32
                ldx     #$7766
                pshd
                jsr     abc
                pshx
test1:          clra
                …
                …
abc:            leas    -2, SP
                pshx
                addd    #$20
                jsr     xyz
                pulx
                leas    2, SP
                rts
                …
                …
xyz:            pshd
test2:          inx
                puld
                rts
```

Solution 9.1:

Shift & test using D
Loop counter in Y
Accumulate count in local variable *count* on stack
Need to save B and Y

```
Stack frame looks like:                 [ return  ]
                                        [address ]
                                        [ save Y ]
                                        [ save B ]
                                SP ⇒    [ count   ]
```

Code is:

```
count:          EQU     0

cnt_one:        pshy                    ; save Y
                pshb                    ; save B
                leas    -1, SP          ; make space for count
                ldd     0, X            ; load data into D
                ldy     #16             ; initialize loop counter

loop:           lsrd                    ; shift LSB into carry
                bcc                     ; test carry for ‘1’
                inc     count, SP       ; increment count
skip:           dbne    Y, loop         ; done yet?
                ldaa    count, SP       ; put result into A
                leas    1, SP           ; release local variable space
                pulb                    ; restore B
                puly                    ; restore Y
                rts
```

Solution 9.2:

(a)	ADDR	DATA
	$5400	??
	$53FF	$32
	$53FE	$8A
	$53FD	[RET]
	$53FC	[ADDR1]
	$53FB	??
	$53FA	??
	$53F9	$66
	$53F8	$77
	$53F7	[RET]
	$53F6	[ADDR2]
	$53F5	$52
SP⇒	$53F4	$8A

(b)	ADDR	DATA
	$5400	??
	$53FF	$32
	$53FE	$8A
	$53FD	$66
SP⇒	$53FC	$77