CSCI-261

Princekumar Diyora
(CSCI user: digpraja)

Test:-2

(a)

(1) false

(2) True

(3) True

(b) In HC11, we can pass the parameters in three ways, direct onto the stack, globally, and direct into the registers. while in SSBC we can only pass the parameter on stack and globally.

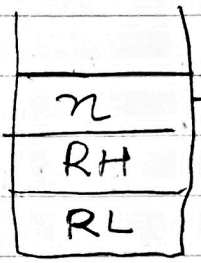
→ In SSBC, before subroutine call we will pass the return addresses and then parameter. now, this way Parameter is on top of the stack and return addresses are underneth of it.

→ so, I think this is really convenient way for accessing the parameter because we don't need to move the return address and also we can take out Parameters.

→ while, in HC11 we can't pass the Parameter that we did in SSBC. in HC11, our subroutine call is in this way that we have to have our parameters underneath of the returnaddress.

→ and because of this if we want to take the parameters then we can access it, underneath of stack.

→ so, in SSBC entry point

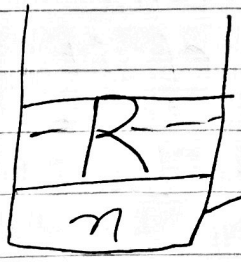


→ take out the parameter

exit point

→ return addresses

→ so, in HC11 entry point



→ take out the parameter underneath of stack

→ exit point

→ return addresses

Q.2

x	y	x or y
0	0	1
0	1	0
1	0	0
1	1	0

(a) N does not set (0 to 1 transition)

```
Pushimm 0
Pushimm 1
Sub
Pushimm ff
nor
Pushext PSW
Popext C
Popinh
halt
```

N flag is zero

```
  01000000
  11111111
  ---------
  00000000
```

↓ n flag

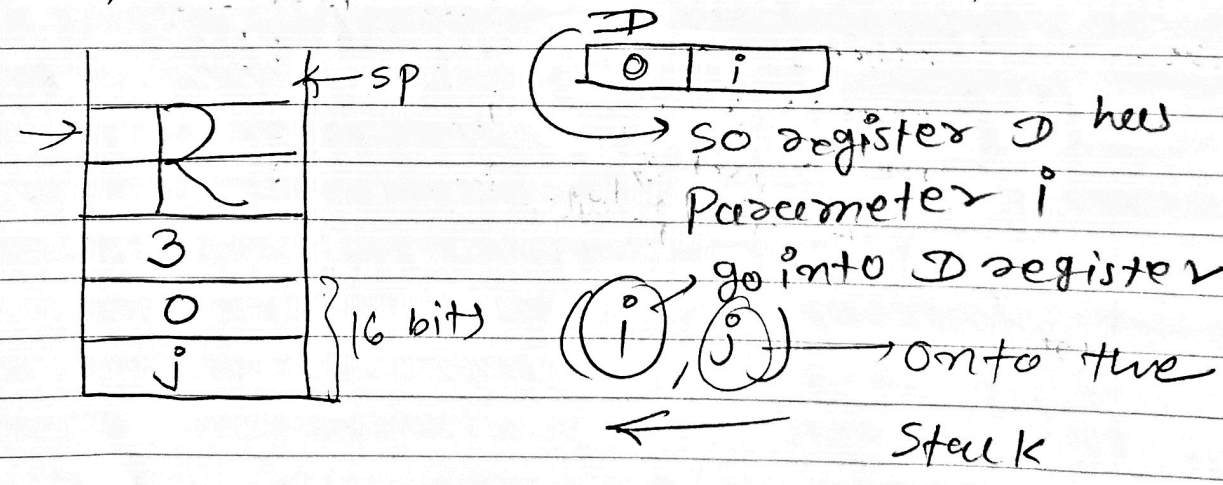
----- C should have the value 0 for n flag

(b) short int foo (short int i, short int j)

return (i+j+3);

}

→ for the short integer parameter passing we pass it into the register D. For parameter passing convention, we will go from right to left. but the first parameter is short int so we will load that into the register D, and then we will pass the second parameter on the stack. Here, short integer will reserve 16 bits space on the stack. Then we will add 3 on top of the stack. so, stack will look like this,



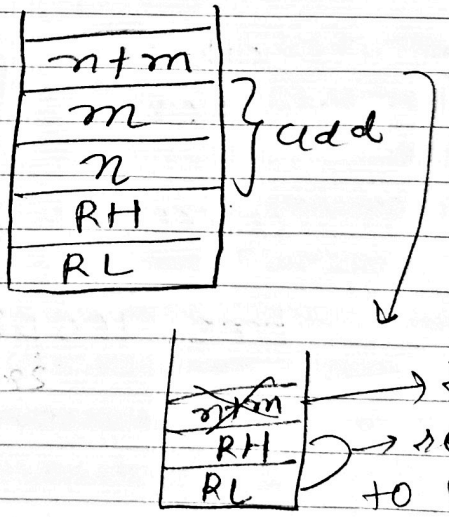
→ and in the end, it will return answer to the D. but for the register D we don't have HC11 instructions so we will use index register X for J.

Q.3

```
        Pushimm 0          } clearing out the flags
        Popext  PSW        } in very beginning

        Pushimm 18 (RL)    -- lower right address
        Pushimm 19 (RH)    -- Higher right address
        Pushext  B         -- take the parameter
                              from Port B and
                              Put it on stack
        Pushext  D         -- take the parameter
                              from port D and
                              Put it on stack
        jnz  add2 (22)     -- jump to the sub-
                              routine call add2.
YY:     Popext  C          -- we are returning
                              value to port C,
        halt

add2:   add
        Popext RL (30)
        Popext RH (31)
        jnz  ReturnAddress
```



Q.4

```
.global sigmanm

sigmanm:
        tpa             ; save ccr
        psha (n)        ; pushing register a on
                          top of the stack
        psha (m)        ; again pushing the
                          register a on top
                          of the stack
        clra            ; clearing the partial
                          sum
        tsta            ; test for n==0
        tstb            ; test for m==0
        beq done        ; will branch to
                          done part

loop:   aba             ; acca = acca + accb
        decb            ; decrementing the
                          b accumulator
        aba             ; again adding accumulator
                          a with the b
        bne loop        ; and then it will
                          if z is non-zero then
                        again go to back to
                        very begening of the
                        loop
```

```
done:   pshu    ; Pushing some In register b
        Pulb
        Pula    ; restoring the ccr
        Pula    ; again
        clra    ; clear the register a
        rts     ; return to the
                  subroutine
```