

This print-out should have 22 questions, check that it is complete. Multiple-choice questions may continue on the next column or page: find all choices before making your selection. The due time is Central time.

EE319K Chapter 9 homework.

001 (part 1 of 1) 10 points

All but one of the following programs violates the standard usage of the stack. For these situations, you may assume the 6812 allocated stack area is **\$0A00** to **\$0BFF** and the stack pointer is currently equal to **\$0B00**. Which of the following illustrates a proper use of the microcomputer stack?

- ```
func psha save
 ldab 0,s read local
 rts
```
- ```
func  psha                save
      ins                balance stack
      ldab -1,s          read local
      rts
```
- ```
func staa -1,s save
 des allocate
 pulb deallocate
 rts
```
- ```
func  leas -3,s          allocate
      staa 2,s           save
      leas 3,s           deallocate
      rts
```
- ```
func leas -3,s allocate
 staa -2,s save
 leas 3,s deallocate
 rts
```

**6.**

```
func staa $0B20,s save
 ldab $0B20,s read local
 rts
```

---

**002** (part 1 of 1) 10 points

Are parameters passed **call by value** or **call by reference** in Java?

- I don't know?
- Because pointers are not allowed in Java, all parameters are passed call by value.

---

**003** (part 1 of 2) 0 points

Consider the following main program that passes one 8-bit parameter to a subroutine using the stack. The subroutine reverses the bits. **n** is the 8-bit input parameter, and **cnt** is the 8-bit local variable.

```
main lds #$0c00 init stack
 ldaa #47
 psha parameter n
 jsr bitrev call sub
 leas 1,s balance stack
loop bra loop
n set nnn binding
cnt set ppp binding
bitrev pshx save old frame
 movb #8,1,-s allocate cnt=8
 tsx create frame
bloop lsr n,x shift n
 lsla shift into A
 dec cnt,x loop counter
 bne bloop
 txs
 pulx
 rts
```

What value should you use in the **nnn** position to implement the proper binding of parameter **n**?

---

**004** (part 2 of 2) 0 points

What value should you use in the **ppp** position to implement the proper binding of local variable **cnt**?

---

**005** (part 1 of 1) 10 points

Which 6812 assembly program calls the subroutine, sub, 525 times?

- ```

1.      ldaa #525      initialize counter
loop   jsr  sub       call the subroutine
      dbne A,loop     decrement loop counter

```
- None of these choices is correct.
- ```

3. ldy #$525 initialize counter
loop jsr sub call the subroutine
 dbne Y,loop decrement loop counter

```
- ```

4.      ldx  #525    initialize counter
loop   jsr  sub       call the subroutine
      dbne X,loop     decrement loop counter

```
- ```

5. ldab #525 initialize counter
loop jsr sub call the subroutine
 dbeq B,loop decrement loop counter

```

---

**006** (part 1 of 1) 10 points

What is the difference between the b1t and 1b1t instructions?

- None of these choices is correct.
  - b1t uses 8-bit PC-relative addressing while 1b1t uses 16-bit absolute addressing.
  - b1t uses 8-bit PC-relative addressing while 1b1t uses 16-bit PC-relative addressing.
  - 1b1t runs faster and takes fewer memory bytes than b1t.
  - They are two names for the same thing.
  - b1t is an arithmetic compare while 1b1t is a logical compare.
- 

**007** (part 1 of 1) 10 points

Consider the situation where a 11 bit parameter is added to a 6 bit parameter. In particular, assume the number  $n1$  can be any value from 0 to 2047, and the number  $n2$  can be any value from 0 to 63. If  $n3 = n2 + n1$ , how many bits are required to store the number  $n3$ ?

Answer in units of bits.

---

**008** (part 1 of 1) 0 points

What is the basic operation of the ediv instruction?

- 16-bit **RegY** divided by 16-bit **RegX** yielding a 16-bit quotient stored in **RegY**, and a 16-bit remainder stored in **RegD**.
  - 32-bit **RegsY:D** divided by 16-bit **RegX** yielding a 16-bit quotient stored in **RegD**, and a 16-bit remainder stored in **RegY**.
  - 32-bit **RegsY:D** divided by 16-bit **RegX** yielding a 16-bit quotient stored in **RegY**, and a 16-bit remainder stored in **RegD**.
  - 32-bit **RegsY:D** divided by 16-bit **RegX** yielding a 16-bit quotient stored in **RegY**, and a 16-bit remainder stored in **RegD**.
  - 32-bit **RegsD:Y** divided by 16-bit **RegX** yielding a 16-bit quotient stored in **RegY**, and a 16-bit remainder stored in **RegD**.
  - None of these choices is correct.
- 

**009** (part 1 of 1) 10 points

What is the basic operation of the emul instruction?

- None of these choices is correct.
- 16-bit **RegD** multiplied by 16-bit **RegX** yielding a 16-bit product stored in **RegD**.
- 16-bit **RegD** multiplied by 16-bit **RegY** yielding a 32-bit product stored in **RegsX:Y**.

4. 16-bit **RegD** multiplied by 16-bit **RegY** yielding a 16-bit product stored in **RegD**.

5. 16-bit **RegD** multiplied by 16-bit **RegY** yielding a 32-bit product stored in **RegsD:Y**.

6. 16-bit **RegD** multiplied by 16-bit **RegY** yielding a 32-bit product stored in **RegsY:D**.

---

**010** (part 1 of 1) 10 points

What is the difference between the **emul** and **emuls** instructions?

1. **emul** is integer multiply while **emuls** is a floating-point multiply.

2. None of these choices is correct.

3. **emul** is an 8-bit multiply while **emuls** is a 16-bit multiply.

4. **emul** operates on signed integers while **emuls** operates on unsigned integers.

5. **emul** is integer multiply while **emuls** is a fixed-point multiply.

6. They are two names for the same thing.

---

**011** (part 1 of 1) 10 points

Consider the situation where a 9 bit parameter is multiplied by a 5 bit parameter. In particular, assume the number  $n1$  can be any value from 0 to 511, and the number  $n2$  can be any value from 0 to 31. If  $n3 = n2 * n1$ , how many bits are required to store the number  $n3$ ?

Answer in units of bits.

---

**012** (part 1 of 1) 10 points

How many bytes are pushed on the stack by the **swi** instruction?

Answer in units of bytes.

---

**013** (part 1 of 1) 10 points

How is the location of the **swi** handler deter-

mined?

1. The **swi** handler is located at address **\$FFF6**.

2. There is a 16-bit pointer to the **swi** handler is located at address **\$FFF6**.

3. None of these choices is correct.

4. The **swi** instruction uses direct addressing.

5. The **swi** instruction uses indexed addressing.

6. The **swi** instruction uses extended addressing.

---

**014** (part 1 of 1) 10 points

Which instruction is used to return from a **swi** interrupt?

1. **rts**.

2. **rtc**.

3. **swi**.

4. None of these choices is correct.

5. **trap**.

6. **rti**.

---

**015** (part 1 of 1) 10 points

Consider a matrix with 5 rows and 7 columns, stored in column-major zero-index format. Each element is 1 byte or 8 bits. Which equation correctly calculates the address of the element at row  $I$  and column  $J$ ?

1. **base+I+J**

2. **base+I+7\*J**

3. None of these choices is correct.

4.  $\text{base} + 7 * \text{I} + \text{J}$

5.  $\text{base} + 5 * \text{I} + \text{J}$

6.  $\text{base} + \text{I} + 5 * \text{J}$

**016** (part 1 of 1) 10 points

Consider a matrix with 6 rows and 4 columns, stored in row-major zero-index format. Each element is 2 bytes or 16 bits. Which equation correctly calculates the address of the element at row I and column J?

1.  $\text{base} + 4 * \text{I} + \text{J}$

2.  $\text{base} + \text{I} + 6 * \text{J}$

3.  $\text{base} + 6 * \text{I} + \text{J}$

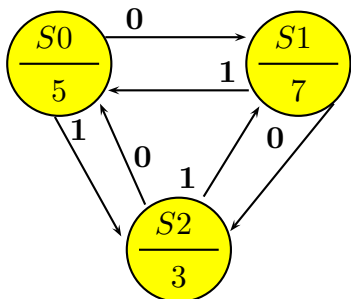
4.  $\text{base} + \text{I} + 4 * \text{J}$

5.  $\text{base} + \text{I} + \text{J}$

6. None of these choices is correct.

**017** (part 1 of 3) 10 points

The following figure shows **Moore** finite state machine. There are three states named **S0**, **S1** and **S2**. There is one input signal, connected to PA0, which can be high or low. There is an 8-bit output, connected to PORTB.



The following program is supposed to implement this **Moore** finite state machine.

```

 org $F000
S0 fcb 5 Output
 fcb 1,2 next states
S1 fcb 7 Output
 fcb 2,0 next states
S2 fcb nnn Output

```

```

 fcb 0,1 next states
FSM clr DDRA PA0 is input
 movb #$FF, DDRB outputs
 ldaa #0 State number
run ldx #S0 Points to fsm
 ldab #mmm size of entry
 mul
 leax D,X points to entry
 ldab 0,X output for this state
 stab PORTB
 ldab PORTA Read input
 andb #$01
 abx add 0 or 1
 ldaa ppp,x Next
 bra run
 org $FFFE
 fdb FSM reset vector

```

What number should be placed in the **nnn** position?

1. 4.

2. 2.

3. 1.

4. None of these choices is correct.

5. 0.

6. 3.

**018** (part 2 of 3) 10 points

What value should be placed in the **mmm** position?

1. 2.

2. None of these choices is correct.

3. 3.

4. 1.

5. 5.

6. 4.

**019** (part 3 of 3) 10 points

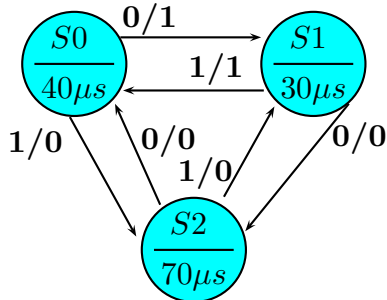
What value should be placed in the **ppp** position?

1. 2.
2. None of these choices is correct.
3. 1.
4. 4.
5. 3.
6. 0.

---

**020** (part 1 of 3) 10 points

The following figure shows **Mealy** finite state machine. There are three states named **S0**, **S1** and **S2**. There is one input signal, connected to PJ0, which can be high or low. There is a 16-bit time for each state, which contains the time to wait in microseconds. There is a 1-bit output, connected to PH0.



The following program is supposed to implement this **Mealy** finite state machine. Notice the sequence of actions performed by this controller is 1)wait, 2) read input, 3) perform output, and 4) go to next state. You may assume the **Wait** subroutine delays for the number of 125ns cycles as specified in RegD.

```

 org $F000
S0 fdb 320 40us wait
 fcb 1,0 outputs
 fcb 1,2 next states
S1 fdb 240 30us wait
 fcb 0,1 outputs
 fcb 2,0 next states
S2 fdb mmm 70us wait

```

```

 fcb 0,0 outputs
 fdb 0,1 next states
FSM lds #$0C00 stack
 bclr DDRJ, #01 PJ0 is input
 bset DDRH, #01 PH0 is output
 ldaa #0 State number
run ldx #S0 Points to fsm
 ldab #nnn size of entry
 mul
 leax D,X points to entry
 ldd 2,X+ time to wait
 jsr Wait
 ldab PORTJ Read input
 andb #01 B is 0 or 1
 abx RegX => Out
 ldaa 0,X Out value
 staa PORTH do output
 ldaa ppp,x Next
 bra run
 org $FFFE
 fdb FSM reset vector

```

What value should be placed in the **mmm** position?

1. 1.
2. 0.
3. 30.
4. 70.
5. None of these choices is correct.
6. 40.

---

**021** (part 2 of 3) 10 points

What value should be placed in the **nnn** position?

1. None of these choices is correct.
2. 4.
3. 2.
4. 0.
5. 6.

**6. 8.**

---

**022** (part 3 of 3) 10 points

What value should be placed in the **ppp** position?

**1. 1.**

**2. 4.**

**3. 3.**

**4. 0.**

**5. 2.**

**6. None of these choices is correct.**