EE319K Final Fall 2005 Solution C.

Jonathan Valvano

(3) Question 1. *Consider a matrix with 4 rows and 6 columns, stored in column-major zero-index format. Each element is 16 bits. Which equation correctly calculates the address of the element at row I and column J?* With column major, the elements in one column are allocated together. The number of elements in each column equals the number of rows (4 in this case). I+4*J gives the element number for (I,J). We add a 2* because each element is 2 bytes and add the base address to get.

H) base+2*I+8*J

(3) Question 2. *Consider the following C program.*

```
short function(const short in){
   return in+5;
}
```

Where is the parameter ***in*** *allocated?* When **const** is added to a parameter or a local variable, it means that parameter can not be modified by the function. It does not change where the parameter is allocated. For example, this example is legal.

```
void LegalFuntion(short in){
  while(in){
    SCI_OutChar(13);
    in--;
  }
}
```

On the other hand, this example is not legal because the function attempts to modify the input parameter. **in** in this example is allocated on the stack or in a register.

```
void NotLegalFuntion(const short in){
  while(in){
    SCI_OutChar(13);
    in--;  // this operation is illegal
  }
}
```

Similarly, this example is not legal because the function attempts to modify the local variable. **count** in this example is allocated on the stack or in a register.

```
void NotLegalFuntion2(void){ const short count=5;
  while(count){
    SCI_OutChar(13);
    count--;  // this operation is illegal
  }
}
```

Most compilers place the first parameter in a register, so either of these answers is correct.

B) local RAM

E) None of these answers is correct

When **const** is a global variable, it does change the way it is allocated. For example, on a microcontroller like the 9S12C32, this variable is defined in ROM (and can not be changed at run time.

```
const short Data=5;
```
On the other hand, this same example on a personal computer like a PC or Macintosh will allocate the parameter in RAM, initialize it when the program is loaded, and prevent the runtime program from changing its value during execution.

(3) Question 3. *What is drop out?* An example of drop out is **F=9*(C/5)+32** (with integer math)
B) Drop out is the error that occurs after a right shift or a divide

(3) Question 4. *Which direction does data flow on the data bus during a write cycle?* A read cycle moves data from memory or input device to the processor. For example, fetch instructions brings machine code from memory into the IR. A write cycle moves data from processor to memory or output device. For example PSHA causes a write cycle, moving data from RegA (processor) to the stack (memory).
A) From 6812 to memory

(3) Question 5. *Which of the following statements best describes the action that will set the RDRF bit in the SCISR1 register on the 6812?* The **RDRF** bit in the **SCISR1** register on the 6812 is set by the hardware when a new incoming serial transmission is received and the data is moved from the receive shift register into the receive data register.
G) None

(3) Question 6. *Which of the following statements best describes the action that will clear the **RDRF** bit in the **SCISR1** register on the 6812?* Clearing the RDRF bit takes two software actions. First the software reads **SCISR1** when **RDRF** is one, then it must read **SCIDRL**.
B) The software reads SCISR1 when RDRF is one, followed by reading SCIDRL.

(3) Question 7. *What event triggers the start of an ADC conversion on the 6812?* Writing to ATDCTL5 starts an ADC conversion.
C) write to ATDCTL5

(3) Question 8. *What is the bug in the following initialized global variable on the 9S12C32?*

```
        org  $3A00
Count   fdb  100
```

The key to this problem is to realize $3A00 is volatile RAM on the 9S12C32. Global variables work differently on an embedded system than they do on a personal computer. When you double-click a **program.exe** file on a personal computer both variables and programs are loaded from secondary storage (e.g., a file on the hard disk) into RAM. This load operation on a personal computer would have initialized the global variable **`Count`**. On an embedded system, programs are burned into ROM and variables are allocated in RAM. Initialized global variables like this will not work properly because the initial value "100" will not exist after power is removed and restored.
B) RAM is volatile

(3) Question 9. *The measurement system range is 0 to 399.9 and a resolution of 0.1. What is the precision in decimal digits?* The precision in alternatives is the range divided by the resolution, 400/0.1 is 4000 alternatives. 1000 alternatives is 3 decimal digits, 2000 alternatives is 3½ decimal digits, and 4000 alternatives is 3¾ decimal digits.

Consider the result of executing the following two 6812 assembly instructions.

```
ldab #101
subb #110
```

(3) Question 10. *What will be the value of the carry (C) bit?* To consider the C bit, convert both numbers to unsigned (0 to 255), perform the subtraction and see if you get the correct answer. 101-110 = -9, which does not fit in an unsigned format, so
C = 1, unsigned answer is incorrect

(3) Question 11. *What will be the value of the overflow (V) bit?* To consider the V bit, convert both numbers to signed (-128 to 127), perform the subtraction and see if you get the correct answer. 101-110 = -9, which does fit in an signed format, so
V = 0, signed result is OK

(3) Question 12. *What will be the value in Register D after executing the following 6812 assembly instructions?*

```
ldd  #100
ldy  #670
emul
```

The **emul** instruction multiplies 16-bit unsigned RegD by 16-bit unsigned RegY to yield a 32-bit unsigned product in Y:D (which means RegY is the most significant 16 bits and RegD is the least significant 16 bits). 100*670=67000. The most significant 16 bits of 67000 will be 1. The bottom 16-bits of 67000 are 67000-65536 = 1464

(3) Question 13. *You may assume all RAM locations are initially 0, and assume Reg Y equals $1234, Reg D is $5678. What is in Reg Y after these instructions are executed?*

```
pshy
pshb
inc 1,s
puly
```

After **pshy**

```
        SP -> $12   big endian
              $34
```

After **pshb**

```
        SP -> $78   RegB is $78
              $12
              $34
```

After **inc 1,s**

```
            SP -> $78
                  $13   inc 1,s modifies memory, not SP
                  $34
```
After **puly**
```
            SP -> $34
```
$7813 is pulled into RegY

(3) Question 14. *An swi pushes the following registers on the stack in this order PC, Y, X, A, B, CCR, with CCR on top. Initially, assume RegX=$4321, RegY=$ABCD, RegD=$8765. What is the resulting hexadecimal value in RegX after these instructions execute?*
```
     stx  5,s
     std  3,s
     rti
```
Initially
```
            SP -> OldCCR
                  OldB
                  OldA
                  MSbyte OldX   big endian
                  LSbyte OldX
                  MSbyte OldY   big endian
                  LSbyte OldY
                  MSbyte OldPC  big endian
                  LSbyte OldPC
```
After **stx 5,s**
```
            SP -> OldCCR
                  OldB
                  OldA
                  MSbyte OldX   big endian
                  LSbyte OldX
                  $43           big endian
                  $21
                  MSbyte OldPC  big endian
                  LSbyte OldPC
```
After **std 3,s**
```
            SP -> OldCCR
                  OldB
                  OldA
                  $87           big endian
                  $65
                  $43           big endian
                  $21
                  MSbyte OldPC  big endian
                  LSbyte OldPC
```
After **rti**

Reg X will be loaded with $8765

(3) Question 15. *Show the machine code generated by the instruction*

```
     orab -5,y
```

$EA $5B

Questions 16 and 17 involve the following assembly code. The subroutine returns the result by value on the stack.

```
main lds #$4000
     leas -2,s   ; make space for out parameter on stack
     jsr GetT
     puly        ; Fetch the return value from the stack
     stop
data set  xxx    ; binding of 16-bit local variable
out  set  yyy    ; binding of 16-bit output parameter
GetT leas -2,s   ; allocate 16-bit local called data
;****body of the subroutine
     ldd  TCNT   ; get time
     std  data,s ; place time into local variable data
     ldd  TCNT   ; get time again
     subd data,s ; time difference in RegD
     std  out,s  ; return by value on the stack
;****end of body
     leas 2,s    ; deallocate data
     rts         ; return
```

(3) Question 16. *What value should you use in the xxx position to implement the binding of the local variable, data?*

```
data set  0
```

(3) Question 17. *What value should you use in the yyy position to implement the binding of the parameter, out?*

```
out  set  4
```

The key to this question is to hand execute the program, and draw a stack picture

After `leas -2,s`

```
            SP -> out
```

After `jsr GetT`

```
            SP -> Return address
                  out
```

After `leas -2,s`

```
            SP -> data
                  Return address
            SP+4  out
```

(3) Question 18. *A SCI is configured at a baud rate of 1200 bits/sec, with 9 bit data, two stop bits, and no parity. What is the bandwidth in bits/sec?*

There are 12 bits/frame, 9 information bits/frame, so 1200*9/12 = 900 bits/sec

(3) Question 19. *Assume the 9S12C32 sequence length is configured to perform one ADC sample (S8C S4C S2C S1C in ATDCTL3 is set to 0001). After the ADC is triggered*

to sample channel 5, into which ATD I/O register is the digital conversion stored? In other words, out of which location does the software read the ADC result?
The first result goes in ATDDR0

(3) Question 20. *A signed 8-bit binary fixed-point number has a resolution of 1/16 = 2-4. If the integer value stored in memory is $F0, what value does it represent?*
First convert to signed integer $F0 equals -128+64+32+16 = -16. The value of the binary fixed point number is integer*resolution = -16 * 1/16 = -1.00

(3) Question 21. *Which term best describes an interfacing method that the software checks the status of an I/O device, and proceeds once the device is ready?*
Gadfly or busy-wait

(3) Question 22. *Which data structure has the following features? It can hold a variable number of fixed-size elements. It has two main operations, one to store data into itself, and a second operation to remove data. The data is removed in a "first come first served" order.*
FIFO queue

(3) Question 23. *Assuming the variable, N, has an 8-bit signed value, does the following operation sometimes give the incorrect answer? Answer **BUG** if there are some values of N, such that the program yields the incorrect answer or NoBUG if the program always gives the correct result.*

```
     ldab  N
     sex   b,x  ;promote to 16-bits
     leax  10,x ;16-bit add RegX=RegX+10
     tfr   x,b  ;demote
     stab  N
```

Bug, 125+10 will overflow (doesn't fit in 8-bit signed number -128 to +127)

Consider the following assembly subroutine that creates two local variables, called p and q. The variable p is 8-bits and initialized to 50. The variable q is 16-bits and initialized to 500. The local variable binding is created using the set pseudo-ops.

```
p    set   xxx          ; binding
q    set   yyy          ; binding
sub1 pshy               ; save register Y
     tsy                ; stack frame
     movb #50,1,-sp     ; allocate and initialize p
     movw #500,2,-sp    ; allocate and initialize q
;... stuff
     ldaa p,y  ; read from p
     ldx  q,y  ; read from q
;... more stuff
     leas 3,s     ; deallocate p,q
     puly         ; restore register Y
     rts          ; return
```

The key to this question is to hand execute the program, and draw a stack picture

After **pshy tsy**

```
            Y -> MSbyte oldY <- SP
                 LSbyte oldY
```

After **movb #50,1,-sp**

```
                 50          <- SP
            Y -> MSbyte oldY
                 LSbyte oldY
```

After **movw #500,2,-sp**

```
          Y-3    MSbyte 500  <- SP
                 LSbyte 500
          Y-1    50
            Y -> MSbyte oldY
                 LSbyte oldY
```

(3) Question 24. *What value should you use in the xxx position to implement the binding of p?*

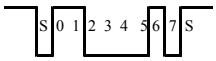
```
p    set   -1
```

(3) Question 25. *What value should you use in the yyy position to implement the binding of q?*

```
q    set   -3
```

(3) Question 26. *Sketch the output waveform occurring on the PS1=TxD output as one character (ASCII 'C' = $43) is transmitted. Assume 8 bit data, no parity and 1 stop bit.*

First write $43 in binary = 01000011. Next add the start bit (0) and a stop bit (1).



(3) Question 27. *In order to observe the where and when our software is executing, we can connect unused output pins to an oscilloscope and set/clear these pins at various important locations within our software. What is this debugging process is called? Choose from stabilization, profiling, desk checking, or dump.*

profiling

(3) Question 28. *Assume PTT is an 8-bit input and PTM is an output. The goal of this code is to clear PTM if PTT bit 0 is set. Which op code should be used in the ??? position?*

```
    ldaa PTT
    anda #1
    ???  skip
    clr  PTM
skip
```

If the bit is 0, then the anda instruction will yield 0 and set the Z bit. The **beq** instruction will skip the clr PTM if PTT bit 0 is clear.

(8) Question 29. This is example **Tree.rtf** within TExaS

```
null     equ  0       Null pointer
Letter   equ  0       Index for command letter
SubAddr  equ  1       Index for subroutine address
Left     equ  3       Index for left next cmd entry
Right    equ  5       Index for right next cmd entry
         org  $4000
Main     lds  #$4000
Exec     jsr  SCI_InChar       Input a char in RegA
         jsr  SCI_OutChar      Echo
         ldx  Root        First entry Command table
Test     cpx  #null
         beq  Exec        Not found, start over?
         cmpa Letter,X    Is it this command?
         beq  Found
         bhi  goRight
goLeft   ldx  Left,X      RegA < Letter
         bra  Test
goRight  ldx  Right,X     RegA > Letter
         bra  Test
Found    ldx  SubAddr,X   X points to the subroutine
         jsr  0,X         Execute command
         bra  Exec
         org  $FFFE
         fdb  Main
```

(8) Question 30. Show the assembly main program

```
Main movb #$3F,DDRM  ; PTM is output
     clr  DDRT       ; PTT is input
     ldab #0         ; I=initial state
loop ldaa #32        ; each state is 32 bits
     mul             ; RegB=32*I
     ldaa PTT        ; 1) input
     anda #$0F
     ldx  #S0        ;base
     abx             ;base+32*I
     leax A,X        ;base+32*I+input
     ldaa 0,x
     staa PTM        ;2) output
     ldab 16,x       ;3) next
     bra  loop
```