First:________________   Middle Initial: _____   Last:____________________

This is a closed book exam. You must put your answers in the space provided. You have 3 hours, so allocate your time accordingly. Please read the entire exam before starting.

(4) Question 1. An embedded system will use an ADC to measure a parameter. The measurement system range is 0.0 to 199.9 and a resolution of 0.1. What is the smallest number of ADC bits that can be used?

(4) Question 2. An 8-bit ADC (different from the 9S12C32) has an input range of 0 to +2.5 volts and an output range of 0 to 255. What digital value will be returned when an input of +0.625 volts is sampled?

(2) Question 3. Consider the result of executing the following two 6812 assembly instructions:
   ldaa #160
   suba #140
What will be the value of the carry (C) bit?

(2) Question 4. Consider the result of executing the following two 6812 assembly instructions:
   ldaa #-90
   adda #-40
What will be the value of the overflow (V) bit?

(4) Question 5. A signed 16-bit binary fixed-point number system has a $\Delta$ resolution of 1/256. What is the corresponding value of the number if the integer part stored in memory is 1152? (hint: 1152=1024+128)
For questions 6, 7
A) Software performs a read SCISR1 with bit set followed by read SCIDRL
B) Software performs a read SCISR1 with bit set followed by write SCIDRL
C) Hardware sets it when there is data in the receive shift register
D) Hardware sets it when there is data in the receive data register
E) Hardware sets it when there is no data in the receive shift register
F) Hardware sets it when there is no data in the receive data register
G) Hardware sets it when there is data in the transmit shift register
H) Hardware sets it when there is data in the transmit data register
I) Hardware sets it when there is no data in the transmit shift register
J) Hardware sets it when there is no data in the transmit data register

(4) Question 6. What sets the TDRE bit in the SCI? Choose a letter A-J

(4) Question 7. What sets the RDRF bit in the SCI? Choose a letter A-J

(4) Question 8. Assume the ADC sequence length is 3 (ATDCTL3 equals $18$) and $85$ is written into ATDCTL5. What happens?
A) Channel 5 is sampled and the result is placed in ATDDR0
B) Channel 5 is sampled and the result is placed in ATDDR5
C) Channel 5 is sampled three times and the results are placed in ATDDR0-ATDDR2
D) Channel 5 is sampled three times and the results are placed in ATDDR5-ATDDR7
E) Channels 5, 6, 7 are sampled and the results are placed in ATDDR0-ATDDR2
F) Channels 5, 6, 7 are sampled and the results are placed in ATDDR5-ATDDR7

(4) Question 9. Which three events cause an interrupt to occur?
Specify three letters in any order.
A) The software disarms the interrupt (e.g., RTIE=0)
B) The I bit in the CCR is set
C) The I bit in the CCR is clear
D) The software arms the interrupt (e.g., RTIE=1)
E) The software acknowledges the interrupt, clearing the flag (e.g., RTIF=0)
F) The software sets the flag bit (e.g., RTIF=1)
G) The hardware sets the flag bit (e.g., RTIF=1)
H) The hardware acknowledges the interrupt, clearing the flag (e.g., RTIF=0)

(4) Question 10. Assuming the variables are 16-bit integers, and all operations are 16-bit integer functions, what error might occur in the following operation? The goal is to multiply N times 0.123 and store the result into M.

\[ M = \frac{(123 \times N)}{1000} \]
A) dropout
B) floor
C) overflow
D) promotion
E) demotion
F) no error can occur because M will be less than N

(4) Question 11. What is the machine code for the following instruction?

```
std 5, sp
```
(5) Question 12. Give the simplified memory cycles produced when the following one instruction is executed. Assume the PC contains $4000, Register X contains $3900, Register A contains $45 and Register B is $67. Just show R/W=Read or Write, Address, and Data for each cycle. Memory locations $3900 through $390F contain $00 to $0F respectively.

$4000 6C31 std 2, x+

<table>
<thead>
<tr>
<th>R/W</th>
<th>Addr</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5) Question 13. You are given an LED with a 2V 20mA operating point. Interface this LED to the 9S12C32, such that the LED is on when PM0 is high (5V) and the LED is off when PM0 is low (0V). The output low voltage of the 7405 is 0.5V. Label all resistor values. No software is required.

(5) Question 14. Interface this switch to the 9S12C32, such that PM1 is high (5V) if the switch is not pressed and PM1 is low (0V) if the switch is pressed. You do not need to debounce the switch. Label all chip numbers and resistor values. No software is required.
(15) Question 15. In this problem you will implement two unsigned 16-bit local variables on the stack using register Y stack frame addressing and symbolic binding. Call one variable front and the other back. The code in this question is part of a subroutine, which ends in rts.

Part a) Show the assembly code that saves RegY, setups up RegY to point into the stack, and allocates the two 16-bit local variables.

Part b) Assume the stack pointer is equal to $3F00, and then this subroutine is called. Draw a stack picture showing the return address, the two variables, RegY and the SP. ($3800 is towards the top, and $3FFF is towards the bottom of the picture). Shade elements that are on the stack.

Part c) Show the symbolic binding for front and back.

Part d) Show code that explicitly implements back=2000;

Part e) Show code that explicitly implements front = 2*back;

Part f) Show the assembly code that deallocates the two 16-bit local variables, and restores Y.
(15) Question 16. Implement the following one-input four-output Mealy finite state machine. The input is on Port T bit 0 and the output is on Port M bits 3,2,1,0. The initial state is happy. You do not need to show the stack initialization or the reset vector.
Part a) Show the ROM-based FSM data structure

Part b) Show the initialization and controller software. Initialize the direction registers, making all code friendly. You may add variables in any appropriate manner (registers, stack, or global RAM). The repeating execution sequence is …input, output (depends on the input and state), next (depends on the input and state)… . Please make your code that accesses Port M friendly.
(15) Question 17. Design a software system that uses the RTI periodic interrupt to create the following repeating waveform on PT7 output.

```
PT7  5.12ms  10.24ms  5.12ms  10.24ms
```

The RTI vector is located at $FFF0. The reset vector is located at $FFFE. For example, a value of $40 written to RTICTL will specify a 1.024ms interrupt period. Show all the software for this system: direction registers, global variables, stack initialization, RTI initialization, main program, RTI ISR, RTI vector and reset vector. The main program initializes the system, then executes a do-nothing loop. The RTI ISR performs output to Port T. Please make your code that accesses Port T friendly. Variables you need should be allocated in the appropriate places.
aba  8-bit add RegY=RegY+RegB
dab  8-bit add with carry to RegA
dbc  8-bit add with carry to RegB
dda  8-bit add to RegA
add  8-bit add to RegB
addb 8-bit add to RegB
addx 8-bit add to RegX
adddx 16-bit add to RegD
adand 8-bit decrement logical and to RegA
adandb 8-bit decrement and to RegB
adcb  8-bit decrement and to RegA
adca 8-bit decrement and to RegB
adcs 16-bit decrement and to RegA
adcsd 16-bit decrement and to RegD
addcx 16-bit decrement add with carry and branch if result≠0
addcxo 16-bit decrement add with carry and branch if result=0
adcc 8-bit logical decrement and complement to RegC
adccs 16-bit logical decrement and complement to RegC
adce 8-bit logical decrement and branch if result≠0
adceo 8-bit logical decrement and branch if result=0
decb 8-bit decrement RegB
dec  8-bit decrement memory
decc 8-bit decrement RegC
decb 8-bit decrement RegB
decss 16-bit decrement RegD
decsss 16-bit decrement RegD
deca 8-bit decrement RegA
deca 8-bit decrement RegA
decss 16-bit decrement RegA
decsss 16-bit decrement RegA
decy 16-bit decrement RegY
decx 16-bit decrement RegX
decs 16-bit decrement RegD
decss 16-bit decrement RegD
decsr 8-bit decrement RegR
decssr 16-bit decrement RegR
decsx 16-bit decrement RegX
decssx 16-bit decrement RegX
decsy 16-bit decrement RegY
decssy 16-bit decrement RegY

orcc  8-bit logical or to RegCC
psha push 8-bit RegA onto stack
pshb push 8-bit RegB onto stack
pshc push 8-bit RegCC onto stack
pshd push 16-bit RegD onto stack
pshx push 16-bit RegX onto stack
pshy push 16-bit RegY onto stack
pula pop 8 bits off stack into RegA
pulb pop 8 bits off stack into RegB
pulc pop 8 bits off stack into RegCC
puld pop 16 bits off stack into RegD
pulx pop 16 bits off stack into RegX
puly pop 16 bits off stack into RegY

rev  Fuzzy logic rule evaluation
revw weighted Fuzzy rule evaluation
rol  8-bit roll shift left Memory
rola 8-bit roll shift left RegA
rolb 8-bit roll shift left RegB
ror  8-bit roll shift right Memory
ror a 8-bit roll shift right RegA
ror b 8-bit roll shift right RegB
rtc return sub in expanded memory
rti return from interrupt
rts return from subroutine
sba  8-bit subtract RegA-RegB
sbca 8-bit sub with carry from RegA
sbcb 8-bit sub with carry from RegB
sec set carry bit, C=1
sei set I=1, disable interrupts
sev set overflow bit, V=1
sex sign extend 8-bit to 16-bit reg
sex B,D
staa 8-bit store memory from RegA
stab 8-bit store memory from RegB

Freescale 6812 addressing modes

<table>
<thead>
<tr>
<th>Pseudo op</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>org</td>
<td>Specific absolute address to put subsequent object code</td>
</tr>
<tr>
<td>=</td>
<td>equ</td>
</tr>
<tr>
<td>set</td>
<td>=equ</td>
</tr>
<tr>
<td>dc.b</td>
<td>db fcb</td>
</tr>
<tr>
<td>fcc</td>
<td>.byte</td>
</tr>
<tr>
<td>dc.w</td>
<td>dw fdb</td>
</tr>
<tr>
<td>dc.l</td>
<td>dl .long</td>
</tr>
<tr>
<td>ds</td>
<td>ds.b rmb</td>
</tr>
<tr>
<td>ds.w</td>
<td>.blkw</td>
</tr>
<tr>
<td>ds.l</td>
<td>.blkl</td>
</tr>
</tbody>
</table>
Address | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | Name
--- | --- | --- | --- | --- | --- | --- | --- | --- | ---
$0240$ | PT7 | PT6 | PT5 | PT4 | PT3 | PT2 | PT1 | PT0 | PTT
$0242$ |DDR7 | DDR6 | DDR5 | DDR4 | DDR3 | DDR2 | DDR1 | DDR0 | DDR
$0250$ | PM7 | PM6 | PM5 | PM4 | PM3 | PM2 | PM1 | PM0 | PM
$0252$ | DDRM7 | DDRM6 | DDRM5 | DDRM4 | DDRM3 | DDRM2 | DDRM1 | DDRM0 | DDRM
$0282$ | ADPU | AFFC | AWA1 | ETRIGLE | ETRIGP | ETRIG | ASCIE | ASCIF | ATDCTL2
$0283$ | 0 | S8C | S4C | S2C | S1C | SIC | FIF0 | FRZ1 | FRZ0
$0284$ | SRE8 | SMPL | SMP0 | PRS4 | PRS3 | PRS2 | PRS1 | PRS0 | ATDCTL4
$0285$ | DJM | DSGN | SCAN | MULT | 0 | CC | CB | CA | ATDCTL5
$0286$ | SCF | 0 | ETORF | FIFOR | 0 | CC2 | CC1 | CC0 | ATDSTAT0
$0288$ | CCF7 | CCF6 | CCF5 | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 | ATDSTAT1
$028D$ | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | ATDDIEN
$0270$ | PAD7 | PAD6 | PAD5 | PAD4 | PAD3 | PAD2 | PAD1 | PAD0 | PTAD
$0272$ | DDRAD7 | DDRAD6 | DDRAD5 | DDRAD4 | DDRAD3 | DDRAD2 | DDRAD1 | DDRAD0 | DDRAD

Address | msb | lb | Name
--- | --- | --- | ---
$0090$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ATADDR0
$0092$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ATADDR1
$0094$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ATADDR2
$0096$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ATADDR3
$0098$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ATADDR4
$009A$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ATADDR5
$009C$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ATADDR6
$009E$ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ATADDR7

Addr | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 | Name
--- | --- | --- | --- | --- | --- | --- | --- | --- | ---
$00C8$ | BTST | BSPL | BRLD | SBR12 | SBR11 | SBR10 | SBR9 | SBR8 | SCIBD
$00C9$ | SBR7 | SBR6 | SBR5 | SBR4 | SBR3 | SBR2 | SBR1 | SBR0 | SCICR2
$00CB$ | TIE | TCIE | RIE | ILIE | TE | RE | RWU | SBK | SCISR1
$00CC$ | TDRE | TC | RDRF | IDLE | OR | NF | FE | PF | SCIDRL
$00CF$ | R7T7 | R6T6 | R5T5 | R4T4 | R3T3 | R2T2 | R1T1 | R0T0 | SCIDR

let $RTR6$, $RTR5$, $RTR4$ be $n$, which is a 3-bit number ranging from 0 to 7
let $RTR3$, $RTR2$, $RTR1$, $RTR0$ be $m$, which is a 4-bit number ranging from 0 to 15
RTI interrupt frequency (Hz) = $15625 \times 2^{-n} / (m+1)$
RTI interrupt period (ms) = $0.064 \times (m+1) \times 2^n$

**STD**

Operation: $(A \cdot B) \Rightarrow M \cdot M + 1$
Description: Stores the content of double accumulator $D$ in memory location $M \cdot M + 1$. The content of $D$ is unchanged.

<table>
<thead>
<tr>
<th>Source Form</th>
<th>Address Mode</th>
<th>Object Code</th>
</tr>
</thead>
</table>
| STD opr8a | DIR | $sc\ dd$
| STD opr16a | EXT | $7c\ hh\ 11$
| STD oprnO_ysxp | IDX | $6c\ xb$
| STD oprn0_ysxp | IDX1 | $6c\ xb\ ff$
| STD oprn16_ysxp | IDX2 | $6c\ xb\ ee\ ff$
| STD [O,ysxp] | [D,IDX] | $6c\ xb$
| STD [opr16,ysxp] | [IDX2] | $6c\ xb\ ee\ ff$