Jonathan W. Valvano First:_____ Last:_____
February 25, 2000, 11:00am-11:50am

This is an open book, open notes exam. You must put your answers on these pages only, you can use the back. You have 50 minutes, so please allocate your time accordingly.
Please read the entire quiz before starting.

(45) Question 1. In this problem you will design a period meter using interrupting input capture. The digital input signal is connected to PT7. The range of periods is 100 µs to 100 ms. You do not need to check for overflow, i.e., you may assume the period will be between 100 µs to 100 ms.
(10) Part a) What is the best period measurement resolution that covers the entire range, assuming a 16-bit precision.

(5) Part b) To store period data in memory, you will need a fixed-point number system. What fixed-point format is best? In particular, how would you represent the period 12.346789 ms in memory?

(30) Part b) Modify this program from Chapter 6 so that the period measurement resolution matches part a) and the input is on PT7. Change both the code and comments.

```
// PT1/IC1 input = external signal
// rising edge to rising edge
// resolution = 500ns
// Range = 36 µs to 32 ms,
// no overflow checking
// IC1 interrupt each period,
unsigned int Period;  // units of 500 ns
unsigned int First;   // TCNT first edge
unsigned char Done;   // Set each rising
void Ritual(void){
    asm(" sei");    // make atomic
    TIOS &= 0xFD;   // PT1 input capture
    DDRT &= 0xFD;   // PT1 is input
    TSCR = 0x80;    // enable TCNT
    TMSK2= 0x32;    // 500ns clock
    TCTL4 = (TCTL4&0xF3)|0x04; // rising
    First = TCNT;   // first will be wrong
    Done=0;         // set on subsequent
    TFLG1 = 0x02;   // Clear C1F
    TMSK1 |= 0x02;  // Arm IC1
    asm(" cli");}
#pragma interrupt_handler TIC1handler()
void TIC1handler(void){
    Period=TC1-First;
    First=TC1;   // Setup for next
    TFLG1=0x02;  // ack by clearing C1F
    Done=0xFF;}
#pragma abs_address:0xffec
void (*TC1_vector[])() = { TIC1handler};
#pragma end_abs_address
```

(10) Question 2. In this problem consider this C function, which performs a write followed by read access to a global variable.

```
int r;      // global variable
int tt(int x, int y){
  r=x;      // write to global
  r=r+y;    // read from global
  return r;}
```

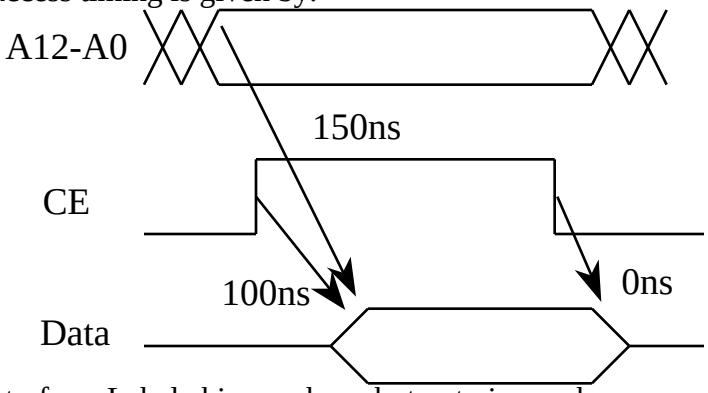
(5) Part a) The following assembly listing was generated by the ICC12 cross-compiler. Is the function reentrant? I added the comments. Give a short justification for your answer. In particular, if you think it is not reentrant, place arrows between pairs of instructions at places where if an interrupt were to occur, data would be lost.

```
ICC12 Version 5
                        ;              y -> 6,x
                        ;              x -> 2,x
 F03B                   _tt::
 F03B  3B                 pshd
 F03C  34                 pshx
 F03D  B775               tfr s,x
 F03F  1805020800         movw 2,x,_r  ; r=x;
 F044  FC0800             ldd _r       ; RegD=r
 F047  E306               addd 6,x     ; RegD=r+y
 F049  7C0800             std _r       ; r=r+y
 F04C  FC0800             ldd _r
 F04F  B757               tfr x,s
 F051  30                 pulx
 F052  1B82               leas 2,sp
 F054  3D                 rts          ; return r in RegD
```

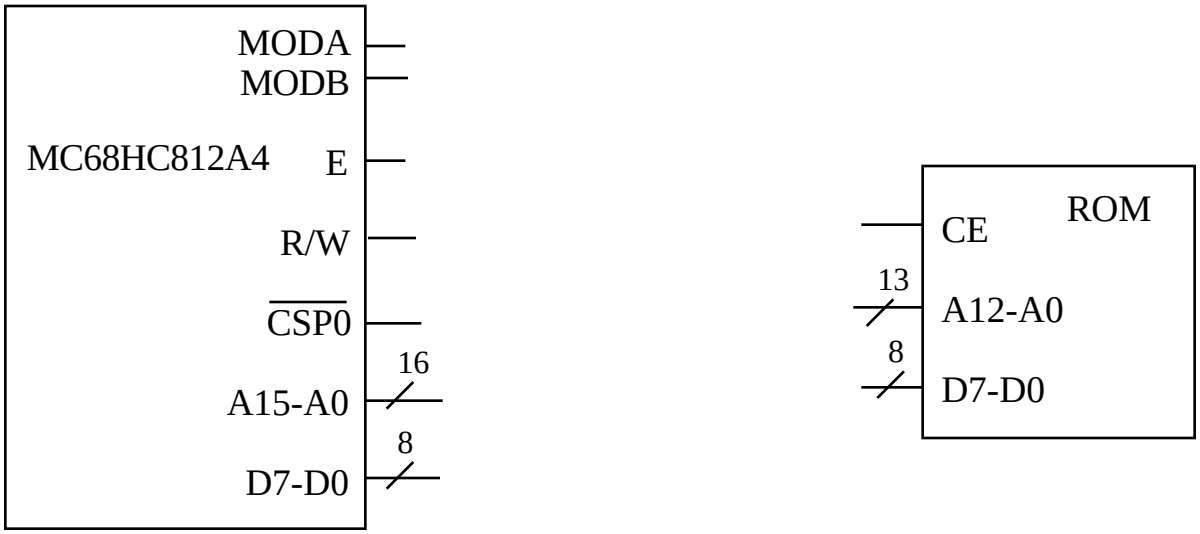
(5) Part b) The following assembly listing was generated by the Hiware cross-compiler. Is the function reentrant? I added the comments. Give a short justification for your answer. In particular, if you think it is not reentrant, place arrows between pairs of instructions at places where if an interrupt were to occur, data would be lost.

```
HI-CROSS+ ANSI-C/cC++ Compiler for HC12 V-5.0.15, Dec 15 1998
Function: tt
  0000 3b          pshd
  0001 ec84        ldd   4,sp  ; RegD=y
  0003 e380        addd  0,sp  ; RegD=x+y
  0005 7c0000      std   r     ; r=x+y
  0008 30          pulx
  0009 3d          rts         ; return r in RegD
```

(45) Question 3. In this problem you will interface an 8192 byte ROM to the MC68HC812A4 running in expanded narrow mode. Use $\overline{\text{CSP0}}$ to place the ROM at $E000 to $FFFF. The 6812 is running at 8 MHz. When the chip enable, CE, is high its data outputs will contain the data at the specified address. The access timing is given by.



(15) Part a) Show the interface. Label chip numbers, but not pin numbers.



(15) Part b) Develop equations for RDA and RDR, and use them to determine the minimum number of cycle stretches required for this interface. Assume a 10ns gate delay.

(15) Part c) Show the combined **read cycle** timing diagram. Show **E**, **R/W**, **A15-A0**, $\overline{\text{CSP0}}$, **CE**, **RDA**, and **RDR**. All signals are outputs except Read Data Required. Use arrows to signify causal relations. Show the timing delays as arrows with numbers in nanoseconds. Calculate the actual RDA and RDR intervals.