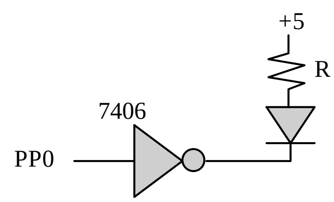
(4) Question 1. Consider DAC parameters.
Part a) Accuracy
Part b) Monotonic
Part c) Resolution
Part d) Precision

(4) Question 2. Write C code that changes the baud rate to 1000 bits/sec.
```
SCI0BD = 625; // n = 8000000/(800*16)
```
(5) Question 3. Use Ohm's Law, V = I*R

3V = R*5V/(10k+R)
30k+3R= R*5
30k = R*2
R = 15k

(6) Question 4. A measurement system has a range of 0 to 19.9 cm and a resolution of 0.1 cm. Only 1 byte is needed.

Part a) Write assembly code that multiplies the position by 0.25 storing the result back into **`position`**.
```
  ldaa location   ;8-bit unsigned fixed point with 0.1 cm resolution
  lsra            ;divide by 2, unsigned
  lsra            ;divide by 2, unsigned
  staa location
```
Part b) Write assembly code that adds 2.0 cm to the variable storing the result back into **`position`**.
```
  ldaa location   ;8-bit unsigned fixed point with 0.1 cm resolution
  adda #10        ;add 1.0
  staa location
```
(4) Question 5. Write a C function that transmits one character.
```
void SCI0_OutChar(char data){
  while((SCI0SR1 & 0x80) == 0){}; // wait for TDRE
  SCI0DRL = data;  // send
}
```
(4) Question 6. R = (5-1-0.5V)/0.01A = 350 Ω



(4) Question 7. Draw stack pictures. Assume RegB = $55, RegX=$1234 and RegY = $5678. What is the value in RegX after executing these instructions?
```
       pshb
SP-> $55
       stx  2,-sp
SP-> $12
     $34
     $55
       sty  1,sp-
SP-> $??
     $56
     $78
     $55
       leas 2,sp
SP-> $78
     $55
       Pulx
```
____$7855___

(6) Question 8. Rewrite the assembly subroutine removing the bug.
```
calc TFR   D,X
     LDY   0,X
     LDD   #314
     EMULS      ;need signed
     LDX   #1000
     EDIVS
     TFR   Y,D
```

```
      RTS
```

(2) Question 9. Consider the result of executing the following two 9S12 assembly instructions.

```
ldaa #156
suba #-50
```

The carry (C) bit will be 1 because **156-206** does not fit in unsigned 8-bit

The overflow (V) bit will be 0 because **-100- -50** does fit in signed 8-bit

(4) Question 10. These six events all occur during each output compare 7 interrupt.

C) 1,3,4,2,5,6

(4) Question 11. Remember to fetch all object code bytes and push the return address on the stack.

R/W	Addr	Data
R	$4005	$16
R	$4006	$42
R	$4007	$00
W	$3FF7	$08
W	$3FF6	$40

(4) Question 12. The 10-bit frame = start,1,0,0,1,1,1,0,1,stop. The data is $B9

(24) Question 13. In this problem you must use a C data structure that stores this Moore FSM.

Part a) Show the C code that defines a linked structure for this FSM.

```
const struct State{
 unsigned char out;        // 1 means on, 0 means off
 unsigned short threshold; // 0.1 F fixed point
 const struct State *next[2];
};
typedef const struct State StateType;
typedef StateType * StatePtr;
#define On  &Machine[0]
#define Off & Machine[1]
StateType Machine[2]={
  {0,700,{Off,On}}, // less than 70 means go to Off
  {1,680,{Off,On}}  // less than 68 means go to Off
};
```

Part b) Write the main that calls **ADC_Init**, initializes the FSM, sets up the OC0, and enables.

```
StatePtr Pt;
void main(void){
  ADC_Init();
  DDRT |= 0x01;    // PT0 output to AC
  Pt = Off;        // initial state
  TIOS |= 0x02;    // activate TC1 as output compare
  TSCR1 = 0x80;    // Enable TCNT, 8MHz
  TSCR2 = 0x07;    // divide by 128, TCNT is 62.5 kHz
  TIE  |= 0x02;    // arm OC1
  TC1   = TCNT+50; // first interrupt right away
asm cli            // enable interrupts
  for(;;){};
}
```

Part c) Write a C function that samples ADC channel 1 using busy-wait synchronization.

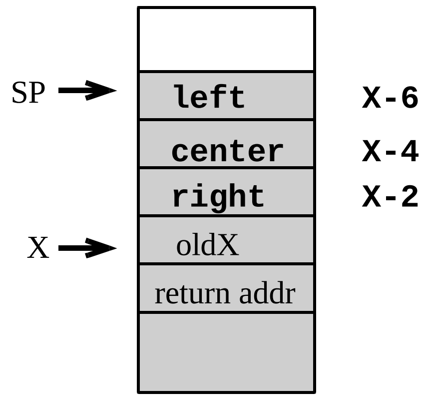
```
unsigned short ADC_In(void){
  ATD0CTL5 = 0x81;                // start sequence
  while((ATD0STAT0&0x80)==0){};    // wait for SCF
  return ATD0DR0;
}
```

Part d) Write the output compare ISR in C that implements the FSM.

```
interrupt 9 void TC0han(void){ unsigned short input;
  input = ADC_In();   // Temperature in 0.1F
  if(input < Pt->threshold){
    Pt = Pt->next[0]; // Next state if input less than threshold
```

```
  } else{
    Pt = Pt->next[1]; // Next state if input greater than threshold
  }
  PTT = Pt->out;      // Output depends on the current state
  TC1 = TC1+6250;     // every 100ms
  TFLG1 = 0x02;       // acknowledge OC1
}
```

(10) Question 14. *Reg X stack frame*



Part a) Saves Register X, establishes the stack frame, and allocates the locals.

```
  pshx
  tsx
  leas -6,sp
```

Part b) Draw a stack picture.

Part c) Show the symbolic binding

```
left   set  -6
center set  -4
right  set  -2
```

Part d) Show code that implements `center=100;` using *Reg X stack frame*.

```
  movw #100,center,x
```

Part e) Show the assembly code that deallocates the local variables, and restores Reg X.

```
  leas 6,sp
  pshx
  rts
```

(15) Question 15. Implement in assembly language a FIFO queue

Part a) Write an assembly subroutine to initialize the FIFO.

```
Fifo_Init clr Size
          rts
```

Part b) Write an assembly subroutine that puts one 16-bit element into the FIFO

```
 Fifo_Put  tfr  d,y
          ldaa Size  ;0,1,2
          cmpa #2
          beq  full
          lsla       ;Reg A is 0 or 2
          ldx  #Buf
          sty  A,X
          inc  Size
          ldd  #0    ;success
          bra  pdone
full      ldd  #1    ;full error
pdone     rts
```

Part c) Write an assembly subroutine that gets one 16-bit element from the FIFO.

```
Fifo_Get  tst  Size      ;0,1,2
          beq  empty
          ldd  Buf       ;get oldest
          std  0,X       ;return by reference
          dec  Size
          movw Buf+2,Buf ;shift data
          ldd  #0        ;success
          bra  gdone
empty     ldd  #1        ;empty error
gdone     rts
```