(10) Question 1. State the term that is described by each definition.
Part a) The process of converting a 16-bit integer into an 8-bit integer is **demotion**.
Part b) Observing 256 different voltage inputs gives **precision** or **8-bit**.
Part c) The part that controls the address and data bus connections to the memory is the **BIU**.
Part d) A scheme that checks a status pin over and over until is called **busy-wait** or **gadfly**.
Part e) Error that can occur as a result of a right shift is **drop out**.
Part f) Error that can occur as a result of a left shift is **overflow**.
Part g) A variable that can be accessed by all functions in the system a **global** or **public** variable.
Part h) A function parameter that contains the data itself is **call by value**.
Part i) A characteristic of a debugger when the presence of the collection of information itself makes a small but unimportant effect on the parameters being measured is called **minimally intrusive**.
Part j) A type of memory that loses its information when power is removed is **volatile**.

(5) Question 2. There are 10 bits per frame and one byte per frame. So the channel bandwidth is 1000 bytes/sec, so this is 1000 samples/sec.

(5) Question 3. Vout = 5V*10kΩ/15kΩ = 3.33 V

(10) Question 4.

```
  tfr  X,D   ;first number, 1/16
  lsrd       ;first number, 1/8
  lsrd       ;first number, 1/4
  pshd       ;save first number
  tfr  Y,D   ;second number, 1/4
  addd 2,sp+ ;sum, balance stack
  bcc  ok
  ldd  #65535
ok
```

(10) Question 5. The first element of the array is the length and remaining are 16-bit signed numbers.
Part a) Write a C function that returns the difference between the maximum and minimum values.

```
short MaxDiff(short *pt){
  short size,max,min,n;
  size = *pt; pt++;
  if(size == 0) return 0;  // empty
  max = -32768; min = +32767;
  while(size){
    n = *pt; pt++;
    if(n > max) max = n;
    if(n < min) min = n;
    size--;
  }
  return max-min;
}
```

Part b) Write an assembly subroutine that performs the same operation.

```
MaxDiff
     leas -4,sp
max  set 0
min  set 2
     tfr  D,X  ;X points to array
     ldd  #0
     ldy  2,X+ ;Y is size
     beq  done
     movw #-32768,max,sp
     movw #32767,min,sp
loop ldd   2,X+  ; value from array
     emaxm max,sp
     eminm min,sp
     dbne  Y,loop
     ldd   max,sp
     subd  min,sp
     leas  4,sp
     rts
```

(5) Question 6. I = (5-2-0.5V)/2500Ω = 2.5V/2500Ω = 1 mA.

(5) Question 7. The answer is ??? because the `std` instruction post decrements over uninitialized RAM, so the `pulx` instruction reads garbage.

(18) Question 8. This question tests your ability to create and use structures.

Part a) Write C code that defines a structure.

```
const struct stuff{
  unsigned char Position[3];  // array of three 8-bit
  unsigned short Time;};      // 16-bit
typedef const struct stuff StuffType;
```

Part b) Define a ROM-based constant with a `Position` of {100,60,50} and a `Time` of 1000.

```
StuffType Command={
  {100,60,50},  // Position
   1000}};      // Time
```

Part c) Set `max` to the largest position number of the three.

```
max = Command.Position[0];
if(max<Command.Position[1]) max=Command.Position[1];
if(max<Command.Position[2]) max=Command.Position[2];
```

Part d) Write a C function that returns the largest position number of the three.

```
unsigned char MaxPosition(StuffType *pt){
unsigned char max;
  max = pt->Position[0];
  if(max< pt->Position[1]) max= pt->Position[1];
  if(max< pt->Position[2]) max= pt->Position[2];
  return max;
}
```

(2) Question 9. **No**, it is not possible for the carry (C) bit to be set. 100 + 100 = 200.

(5) Question 10. 2^2 = 4. TCNT runs at 8 MHz divided by 4 = 2000 kHz. The output compare ISR runs at 2000 kHz divided by ?????, which should be 200 Hz. So ????? is 2000 kHz/0.2kHz = 10000.

(5) Question 11. **`$4003 0750    bsr Function`**

Part a) The return address **$4005** is pushed on the stack during the execution of **`bsr`**.

Part b) PC relative rr=$50, the target address is $4005+$50 = **$4055**.

(20) Question 12. In this problem, your software should output ‘A’ ‘B’ ‘C’ … ‘Z’ over and over

Part a) Show the C code that specifies any global variables you need.

```
unsigned char Letter; // character A to Z
```

Part b) Write the initialization function in C that sets up the SCI0 interrupts. The main will call this initialization once at the beginning, and then perform unrelated tasks. This function should arm and enable interrupts.

```
void SCI_Init(unsigned long baudRate){
  Letter = ‘A’;
  SCIBD = 8000000/16/10000; // br=MCLK/(16*BaudRate)
  SCICR1 = 0;
  SCICR2 = 0x8C;
/* bit value meaning
    7   1    TIE, no transmit interrupts on TDRE
    6   0    TCIE, no transmit interrupts on TC
    5   0    RIE, no receive interrupts on RDRF
    4   0    ILIE, no interrupts on idle
    3   1    TE, enable transmitter
    2   1    RE, enable receiver
    1   0    RWU, no receiver wakeup
    0   0    SBK, no send break */
asm cli   /* enable interrupts */
}
```

Part c) Write the ISR in C that outputs the alphabet using SCI0.

```
interrupt 20 void SciHandler(void){
  if(SCISR1&TDRE){
    SCIDRL = Letter;   // clears TDRE
    if(Letter == ‘Z’){
      Letter = ‘A’;
    } else{
      Letter++;
    }
  }
}
```