Jonathan W. Valvano May 10, 2013, 2-5pm

(20) Question 1. For each term, give a short definition (less than 20 words). Equations are good.

Part a) **Aging** A technique used in priority schedulers that temporarily increases the priority of low priority treads so they are run occasionally.

Part b) **Board Support Package** A set of software routines that abstract the I/O hardware such that the same high-level code can run on multiple computers.

Part c) **Crisp Input** An input parameter to the fuzzy logic system, usually with units like cm, cm/sec, °C

Part d) **External Fragmentation** A condition when the largest file or memory block that can be allocated is less than the total amount of free space on the disk or memory.

Part e) **Internal Fragmentation** Storage that is allocated for the convenience of the operating system but contains no information. This space is wasted.

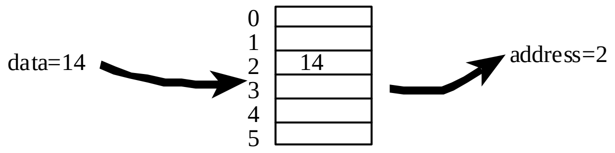
Part f) **Path Expression** A software technique to guarantee subfunctions within a module are executed in a proper sequence. For example, it forces the user to initialize I/O device before attempting to perform I/O.

Part g) **Reentrant** A software module that can be started by one thread, interrupted and executed by a second thread.

Part h) **Stabilization** In the debugging context, we stabilize the problem by creating a test routine that fixes (or stabilizes) all the inputs.

Part i) **Stuff Bits** utilized by inserting a complementary bit after a fixed number of bits of equal value.

Part j) **CAM**. A content addressable memory takes the data and tells you the address at which this data is located.

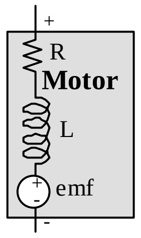


(5) Question 2.

Part a) What is the complex impedance of an ideal inductor? $Z_L = Ls$

Part b) What is the complex impedance of an ideal capacitor? $Z_c = 1/(Cs)$

Part c) Draw an equivalent circuit describing a DC motor and label each of the components.



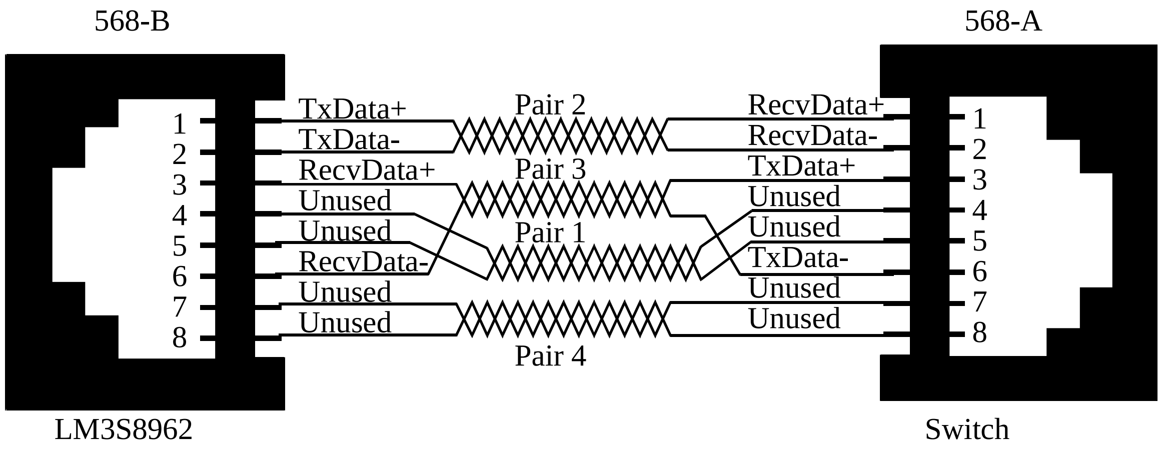
Part d) The R comes from the resistance of the wires in the windings. The L comes from the windings. The wire is wound in coils to produce an electromagnet. The coil-shape creates inductance. The emf comes from the bidirectional electrical to mechanical energy transfer, created from the EM fields.

(5) Question 3. Let user code run with PSP, and let OS code and interrupt service routines run with MSP. Implement all OS calls as software interrupts, causing the OS will run with MSP. This way user bugs are less likely to cause OS crashes.

(5) Question 4. Consider Ethernet at its lowest level

Part a) Ethernet is full-duplex because there is a two pairs of data wires so both directions can occur simultaneously

Part b) Ethernet is asynchronous because there is no clock in the cable.



(5) Question 5. Give the equation defining the Discrete Fourier Transform (DFT). Let x(n) be the sampled data, and make X(k) be the DFT output.

$$X(k) = \sum_{n=0}^{N-1} x(n)\,W_N^{kn} \quad \text{where} \quad W_N = e^{-j2\pi/N} \quad k = 0,1,2,\ldots,N-1$$

(15) Question 6. Write C code for a FIFO queue that can be used to pass 32-bit data between foreground threads. None of the FIFO functions will be called from the background. You must write all of the FIFO code. There can be multiple producers and multiple consumers running in the foreground. You can define and initialize semaphores simply by adding globals:

```
long semaphore=0;
```

You may call the following two blocking semaphore functions without showing their implementation.

```
void OS_Wait(long *semaPt);
void OS_Signal(long *semaPt);
```

Add the `static` qualifier to private components, and no `static` for public components. A producer thread should block on full, and a consumer thread should block on empty.

```
#define FIFOSIZE 16     // can be any size
static long Mutex = 0;                  // implements mutual exclusion
static long DataRoomLeft = FIFOSIZE-1; // size of queue
static long DataAvailable = 0;         // number currently in FIFO
typedef char dataType;
dataType static volatile *PutPt;  // put next
dataType static volatile *GetPt;  // get next
dataType static Fifo[FIFOSIZE];

void Fifo_Init(void){
  OS_Wait(&Mutex); // this is critical
  PutPt = GetPt = &Fifo[0]; // Empty
  DataRoomLeft = FIFOSIZE-1; // size of queue
  DataAvailable = 0;         // number currently in FIFO
  OS_Signal(&Mutex);  // end of critical section
}
void Fifo_Put(dataType data){
  OS_Wait(&DataRoomLeft); // wait for space
  OS_Wait(&Mutex);        // this is critical
  *(PutPt) = data;        // Put
  PutPt = PutPt+1;
  if(PutPt ==&Fifo[FIFOSIZE]){
    PutPt = &Fifo[0];      // wrap
  }
  OS_Signal(&Mutex);          // end of critical section
  OS_Signal(&DataAvailable);  // one more entry
}
void Fifo_Get(dataType *datapt){
  OS_Wait(&DataAvailable);  // wait for data
  OS_Wait(&Mutex);          // this is critical
  *datapt = *(GetPt++);     // return data
  if(GetPt==&Fifo[FIFOSIZE]){
    GetPt = &Fifo[0];       // wrap
  }
  OS_Signal(&Mutex);         // end of critical section
  OS_Signal(&DataRoomLeft);  // more space
}
```

Or we could implement

```
dataType Fifo_Get(void){ dataType data;
  OS_Wait(&DataAvailable);  // wait for data
  OS_Wait(&Mutex);          // this is critical
  data = *(GetPt++);        // get data
  if(GetPt==&Fifo[FIFOSIZE]){
    GetPt = &Fifo[0];       // wrap
  }
  OS_Signal(&Mutex);         // end of critical section
  OS_Signal(&DataRoomLeft);  // more space
  return data;
}
```

(5) Question 7. We defined time-jitter, δt, as the difference between when a periodic task is supposed to be run, and when it is actually run. The goal of a real-time DAS is to start the ADC at a periodic rate, Δt. Let t_n be the nth time the ADC is started. The goal is to make $t_n - t_{n-1} = \Delta t$. The jitter is defined as the constant, δt, such that

$$\Delta t - \delta t < t_i - t_{i-1} < \Delta t + \delta t \qquad \text{for all } i.$$

Assume the input to the ADC can be described as $V(t) = A + B\mathbf{cos}(2\pi ft)$, where A, B, f are constants. Assume the input is not clipped by the finite range of the ADC ($0 < V(t) < 3$V). Derive an estimate of the maximum voltage error, δV_{max}, caused by time-jitter. Basically, solve for δV_{max} as a function of $\delta t, A, B$, and f.

The important concept in this problem is that real-time specification depends on the frequency of the signal. As the frequency increases the time jitter specification must reduce.

Step 1. Find the derivative of $V(t) = A + B\mathbf{cos}(2\pi ft)$

$dV(t)/dt = -2\pi f B\,\mathbf{sin}(2\pi ft)$;units V/sec

Step 2. Find the maximum of the derivative

$|dV(t)/dt_{\text{max}}| = 2\pi f B$;units V/sec

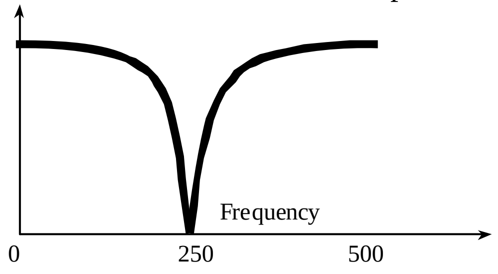
Step 3. Multiply the maximum of the derivative times the time-jitter, δt,

$\delta V = 2\pi f B\, \delta t$;units V

Open book part

(10) Question 7. The sampling rate of a real-time data acquisition system is 1000 Hz.

Part a) We travel along the unit circle, z=$e^{2\pi f/fs}$, a short distance to a zero makes the gain go down, and a short distance to a pole makes the gain go up. Because of the zeros at z = ±j, which is f=±250Hz, this is a digital notch filter. The gain versus frequency plot should go from 0 to ½ f_s. It is a low-Q notch because of the distance to the poles.



Part b)

There are zeros at *z*=±j There are poles at *z*=± ¼ j. The *H(z)* is

$$H(z)=\frac{(z-j)(z+j)}{(z-\frac{1}{4}j)(z+\frac{1}{4}j)}=\frac{z^2+1}{z^2+\frac{1}{16}}$$

Convert to standard form (negative exponents on *z* terms), by dividing top and bottom by *z*

$$H(z)=\frac{1+z^{-2}}{1+\frac{1}{16}z^{2}}$$

Write in terms of *Y(z)* and *X(z)*

$$Y(z)\left(1+\tfrac{1}{16}z^{2}\right)=X(z)\left(1+z^{-2}\right)$$

Take inverse transform to get *y(n)* in terms of *x(n)*

y(n) + 1/16 y(n-2)= x(n)+ x(n-2)

Write in standard format

y(n) = x(n) + x(n-2) - y(n-2)/16

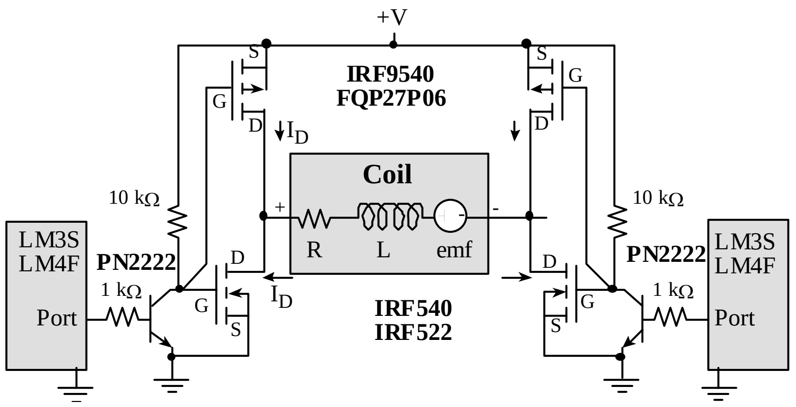
The magnitude at 500 Hz (z = -1) is 2/(1+1/16) = 32/(16+1) = 32/17.

Part c) Write a C function using integer math to implement this digital filter. The function prototype is

```
short Filter(short input);
// input is the new sampled data (0 to 1023), and
// the return parameter is the output of the filter
short static y,y1,y2;  // MACQ length 3
short static x,x1,x2;  // MACQ length 3
  x2=x1; x1=x; x=input;
  y2=y2; y1=y;
  y = x + x2 - y2>>4;
  return y;
}
```

(10) Question 8. With a FAT there is one entry for each block. Let $2^m * 2^n = 2^{29}$, where 2^n is the block size and there are 2^m blocks. For now, we guess the block number is a 16-bit number, so there are 2^{n-1} block numbers in the FAT; so m=n-1, or n=15, m=14. Block size is 32768 bytes; block number is 14 bits wide; and there are 16384 blocks.

(10) Question 9. Design a full H-bridge using two n-channel and two p-channel MOSFETs that will be used to drive a DC motor. Piece together components of Figure 8.11 in the book. The snubber diodes are built into the MOSFETs.



(15) Question 10. PendSV will be used to implement a cooperative thread switcher
Part a) Almost identical to Program 4.9 in the book (swap accesses to Next, SP)

```
PendSV_Handler                 ; 1) Saves R0-R3,R12,LR,PC,PSR
    CPSID   I                  ; 2) Prevent interrupt during switch
    PUSH    {R4-R11}           ; 3) Save remaining regs r4-11
    LDR     R0, =RunPt         ; 4) R0=pointer to RunPt, old thread
    LDR     R1, [R0]           ;    R1 = RunPt
    STR     SP, [R1,#4]        ; 5) Save SP into TCB
    LDR     R1, [R1]           ; 6) R1 = RunPt->next
    STR     R1, [R0,#4]        ;    RunPt = R1
    LDR     SP, [R1]           ; 7) new thread SP; SP = RunPt->sp;
    POP     {R4-R11}           ; 8) restore regs r4-11
    CPSIE   I                  ; 9) tasks run with interrupts enabled
    BX      LR                 ; 10) restore R0-R3,R12,LR,PC,PSR
```

Part b) To invoke a PendSV, the software sets bit 28 of the NVIC_INT_CTRL_R register.

```
void ContextSwitch(void){
  NVIC_INT_CTRL_R = 0x10000000;
}
```