EE 3724 Test #3 Solutions - Fall '00 – Reese

1. (24 pts) For each of the following questions, use one or more of the following terms (they can be used more than once, put ALL applicable terms for an answer):
   SRAM, SSRAM, SDRAM, DRAM, RDRAM, Dual Port SRAM, EEPROM, Flash RAM
   a. What memory types are non-volatile*?* *Flash RAM, EEPROM*
   b. What memory type supports simultaneous access to two memory locations? *Dual Port SRAM*
   c. What memory types are useful for fast, random access caches? *All SRAM types (SRAM, SSRAM).*
   d. What memory types are highest density? *All DRAM types (DRAM, SDRAM, RDRAM)*
   e. What memory uses limited-swing electrical signaling on its interface? *RDRAM*
   f. What memory types support burst transfers using a clock? *Synchronous memories (SSRAM, SDRAM, RDRAM)*

2. (3 pts)Why does DRAM memory contents require periodic refreshing by a DRAM controller?
   The DRAM memory cell consists of 1 transistor + 1 capacitor. The capacitor contents leak over time, and must be periodically refreshed.

3. (3 pts) What does the term 'double date rate' mean when used in conjunction with SSRAM and SDRAMs*?* *Data is transferred on both edges of the clock. Normal SSRAMs, SDRAMS transfer data on only one clock edge*.

4. (4 pts) How many address pins would a 4M x 8 SRAM have? How many address pins would a 4M x 8 DRAM have?
   4 M = 2^{22} = 22 address pins for SRAM. DRAM address pins are multiplexed between row/column, so DRAM only has 11 address pins.

5. (3 pts) Both the NMI and INTR input pins on the x86 are used to signal an external interrupt. What is the principle difference between these two interrupts?
   The NMI is a non-maskable interrupt and cannot be ignored. The INTR can be masked (be ignored) via the IF flag in the condition code register.

6. (3 pts) Give an example of an internally generated interrupt on the x86 (FYI: The software interrupt instruction 'INT' is not a valid answer). *Divide-by-zero, single-step interrupt*.

7. (3 pts) How does the x86 determine the interrupt number for an external INTR interrupt?
   An interrupt acknowledge bus cycle is performed and the interrupting device must place the interrupt number on the lower 8 data lines. The INTA# output pin on the x86 is used to signal that this bus cycle is an interrupt acknowledge cycle.

8. (3 pts) Before the X86 jumps to an interrupt service routine, what gets pushed on the stack?
   CS, IP, Flag register

9. (3 pts) How is the condition code register modified by the X86 before the interrupt service routine is executed? *The IF flag is cleared to zero so that by default other external interrupts will not interrupt the interrupt service routine. The ISR can chose to set the IF flag back to 1 if it wants to allow higher priority interrupts. The TF flag (single step flag) is also set to 0.*

10. (8 pts) For the 8255A, assume the following port locations: Port A: 0520h, Port B: 0524h, Port C: 0528h, Control: 052Ch. Write an instruction sequence that will configure Port A as an INPUT, Port B as an OUTPUT, Lower half of Port C as an INPUT, and Upper Half of port C as an OUTPUT. Use MODE 0 for all ports.

```
 mov  dx, 052Ch
 mov   al, 10010001b
out     dx,al
```

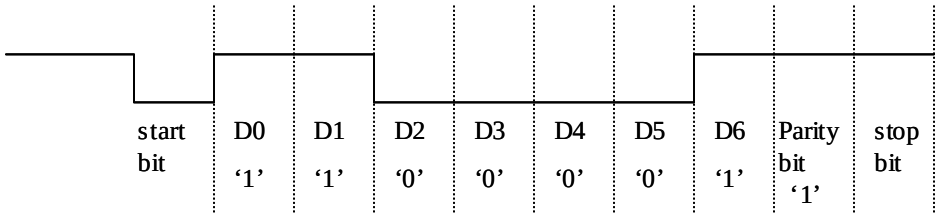
11. (8 pts) For 8255A, and assuming the port definitions above, write an instruction sequence that will read a byte from Port A and then write that byte to Port B.

```
 mov  dx, 0520h
 in     al, dx          ;; read port A
mov   dx, 0524h
out    dx, al          ;; write to port B
```

12. (3 pts) Does the RS-232 port on the PC support half-duplex or full-duplex operation?
Supports full-duplex operation. Has both transmit and receive lines, so can send data in both directions simultaneously.

13. (4 pts) Draw the serial data waveform for sending the byte 43h assuming a data format of 7 data bits, Even parity, 1 stop bit. Data is sent LSB first, use positive logic.

43h = 100 0011 parity bit = 1 since # of '1' bits is odd and we want even parity.



14. (3 pts) What causes an 'overrun' error on the 16550D UART?
When the processor does not read the UART often enough, and the input FIFO fills up, such that incoming characters overwrite characters that are already in the FIFO.

15. (15 pts) For the 16550D, assume the following port definitions:

```
; all port definitions for COM1
rbr         equ   03f8h
thr         equ   03f8h
ier         equ   03f9h
iir         equ   03fAh
fcr         equ   03fAh
lcr         equ   03fBh
mcr         equ   03fCh
lsr         equ   03fDh
msr         equ   03feh
scr         equ   03ffh

dl_lsb      equ   03f8h
dl_msb      equ   03f9h
```

Write a subroutine that will check if data is available, and return the character in the AL register along with setting the carry flag = 1. If no data is available, the subroutine should return immediately with the carry flag set = 0 (you cannot assume the value of the carry flag when the subroutine is called). Your subroutine should NOT wait for data to become available. Use the PORT NAMES above, DO NOT use absolute port numbers.

```
Inchk       proc near
            mov   dx,  lsr        ;;; need to read line status register
            in    al, dx          ;;  read line status register
            test  al,  01h;       ;; check if Data Ready bit (bit 0) is a '1'
            jnz   got_data        ;; branch if we have data
            clc                   ;; clear carry flag and return
            ret
got_data:
            mov   dx, rbr         ;;
            in    al, dx          ;; read receive buffer register
            stc                   ;; set carry flag and return
            ret
Inchk       endp
```

16. (10 pts) For the 16550D, assume the port definitions above. Write a code segment that will set the baud rate to 4800 assuming an 3.072 Mhz crystal, and set the data format to 7 bits, Even parity, 1 stop bit. Use the PORT NAMES above, not the absolute port numbers.

```
mov    dx, lcr
mov    al,  80h
out    dx, al         ;; set DLAB bit = 1 to access divisor registers
mov    al, 28h        ;; divisor = 40 = 28h
mov    dx,  dl_lsb
out    dx, al         ;; write to low byte of divisor latch
mov    al, 0          ;; upper byte of divisor needs to be 0
mov    dx,  dl_msb
out    dx, al         ;; write to high byte of divisor latch
mov    dx, lcr
mov    al,  00011010b   ;; 7 bits, even parity, 1 stop bit,  DLAB bit = 0
out    dx, al
```