EE 3724 Test #2 Solutions - Fall '01 – Reese

Work all problems. Closed book, closed notes; You may use the supplied reference material.
Powers of 2: $1K = 2^{10}$, $1M = 2^{20}$, $2^9 = 512$, $2^8 = 256$, $2^7 = 128$, $2^6 = 64$, $2^5 = 32$, $2^4 = 16$

1. 10 pts. Assume a 80486 processor (32-bit data bus). Mark the value of the byte enable signals as TRUE (asserted) or FALSE (negated) for each of the following memory read operations. BE0 is for the lower 8 data lines (D7-D0), while BE3 is for the upper 8 data lines (D31-D24).
Address values are in HEX!!!

	BE3	BE2	BE1	BE0
a. mov ax, [7FFFC]	*F*	*F*	*T*	*T*
b. mov al, [CFF06]	*F*	*T*	*F*	*F*
c. mov ax, [03A49]	*F*	*T*	*T*	*F*

2. (10 pts) Assume a 80486 processor (32-bit data bus). Mark the following accesses as ALIGNED or MISALIGNED.

a. mov ax, [0ADCB] ALIGNED **MISALIGNED**

b. mov ax, [0ADC1] **ALIGNED** MISALIGNED

b. mov eax, [0ADCC] **ALIGNED** MISALIGNED

3. (10 pts) Assume a Pentium processor with a 64 bit wide data bus.

a. How many CLOCK cycles would it take to transfer 128 bytes using normal read cycles?
8 bytes transferred per read cycle, 128/8 = 16 read cycles, 2 clocks per read cycle, so 32 clocks.

b. How many CLOCK cycles would it take to transfer 128 bytes using burst read cycles?
32 bytes transferred per burst cycle, 128/32 = 4 burst cycles, 5 clocks per burst cycle, so 20 clocks

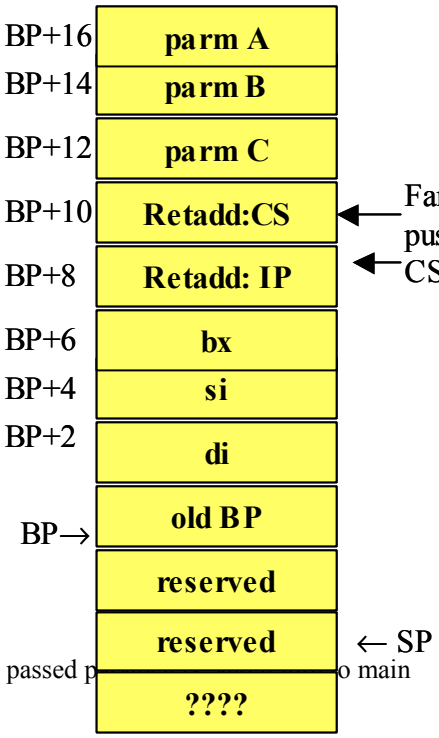
c. How many clock cycles would it take to transfer 128 bytes using pipelined burst read cycles?
32 bytes transferred per burst cycle, 128/32 = 4 burst cycles, 4 clocks per pipelined burst cycle, but first cycle must be non-pipelined burst cycle, so 5 + 3(4) = 17 clocks.

4. (10 pts) Look at the code below and answer the questions

```
MAIN        push  ax          ; push PARAMETER A
            push  bx          ; push PARAMETER B
            push  cx          ; push PARAMETER C
            call  SUBA
            …..

SUBA        proc  far         ;;; FAR subroutine!!!!!!!
             Push bx
             Push si
             Push di
             Enter  4,0
              ;; stack frame has been setup,  POINT A
             ;;; other instructions

             ;; code needed here to clean up stack frame and passed p
SUBA        endp
```



a. At 'point A', write an instruction that will read the value of parameter A into register AX. This instruction cannot change the stack (i.e, a 'POP' is incorrect answer). You must use BP as your index register. It will help if you draw a picture of the stack. BE CAREFUL - SUBA is a 'far' procedure call from MAIN!!!

move ax, [bp+16]

b. Write a TWO INSTRUCTION SEQUENCE at the end of 'SUBA' that will clean up the stack frame, return to the main program, and clean up the stack of the passed parameters **A** and **B**.

As corrected in class, this actually takes more than 2 instructions:

```
leave
pop di
pop si
pop bx
ret  6
```

5. (6 pts) For a memory chip with control lines **CS**, **OE**, and **W**, what control lines must be asserted during:

a. a write operation? *Both CS and W must be asserted*

b. a read operation? *Both CS and OE must be asserted.*

6. (14 pts) Fill in the blanks using one of the following terms. You can also use a number to fill in the blank like 10, 2, 7, etc:

   DRAM RDRAM SRAM SSRAM 'Access time' 'Cycle Time' 'Chip Select' 'non-volatile' 'volatile' ' 'output enable' 'capacitor'

   a. This memory technology has a 6 transistor cell: _____{SRAM,SSRAM}_____

   b. This memory technology has an internal counter to support burst mode: __{SRAM,RDRAM}__

   c. This memory technology has access time = cycle time: __{SRAM,SSRAM}_____

   d. This memory technology uses limited swing signaling technology to achieve high bandwidth: ___RDRAM_____

   e. Time from when address is valid to time when data is valid: __ACCESS TIME_____

   f. # of address pins for a 1M x 1 DRAM: ___10_____

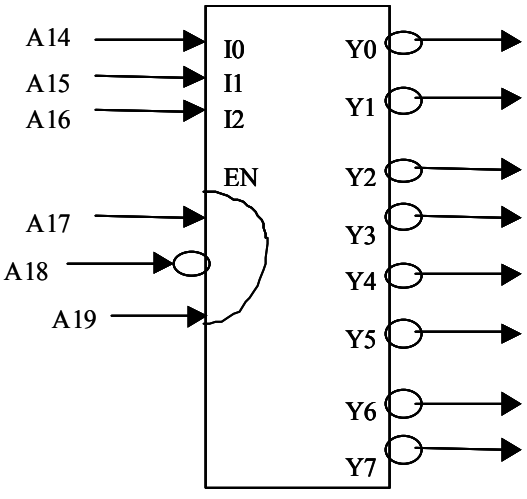
   g. This memory technology has memory cells which need periodic refreshing: _{DRAM,RDRAM}_____

7. (10 pts) For the address decoder below:
   a. Give the range of addresses that output Y6 is valid for (Use HEX addresses)

   b. How many TOTAL bytes is this address decoder valid for? (give the answer in Kbytes, or Mbytes)

A19 A18 A17 A16 A15 A14 A13-A0
1 0 1 1 1 0 0 …. 0 = B8000
1 0 1 1 1 0 1…..1 = BBFFF

A16-A0 = 17 address lines, 2^{17} = 128 Kbytes



3-to-8 Decoder
I2 is MSB, I0 is LSB

8. (10 pts) Write a subroutine that will copy the NULL terminated string pointed to by register SI to the location pointed to by DI. During the copy operation, ONLY copy the letters 'A'-'Z', and 'a'-'z' – ignore any ASCII codes outside of these ranges. You must copy the NULL terminator as well. On entry to the subroutine, SI points to the string to be copied, and DI points to the destination address.

```
scopy       proc    near
            mov     al,[si]         ;get a byte
            cmp     al,0            ;check if at end
            je      exit            ;exit if at end
            cmp     al, 'A'
            jl      skip            ; if lower than 'A', skip
            cmp     al,'Z'
            jle     docopy          ; if between 'A'-'Z', copy
            cmp     al,'z'
            ja      skip            ;if higher than 'z' skip
            cmp     al,'a'
            jb      skip            ;if lower than 'a', skip
docopy:     mov     [di],al         ;copy it
            inc     di              ;increment destination pointer
skip:       inc     si              ;increment source pointer
            jmp     scopy           ;do it again
exit:       mov     [di],al         ;copy null
            ret
scopy       endp
```

9. (10 pts) Write a subroutine that will write the value in AL to the screen in ASCII binary format (i.e., if AL = A3h, then "10100011" would appear on the screen. You can only use the DOS INT 21h single character output function (AH =02, DL has character to output).

```
pbin        proc    near
            mov     cx,8            ;need to print 8 digits
lp1:        shl     al,1            ;shift MSB into carry flag
            jc      do_one          ;if C=1, then print a '1'
            mov     dl, 30h         ;print a '0'
            jmp     print           ;
do_one:     mov     dl, 31h         ;get a '1'
print:      mov     ah,02           ; print digit in dl
            int     21h
            loop    lp1             ;loop until 8 digits printed
            ret
```

10. (10 pts) Write a subroutine called 'memdump' that will print 16 bytes starting at DS:SI to the screen in the format used by the "DEBUG" program as shown below:

```
C:\users>debug

0AC3:0100  32 DB 86 1C E8 39 EB 3B D6 73 1B 56 51 8B CE 8B   2....9.;.s.VQ...
```

Note that the first thing printed is the hex value of DS:SI, followed by hex values of the 16 bytes starting at [DS:SI]. After that, the ASCII representation of the 16 bytes is printed. If the ASCII representation is a non-printable character (ASCII value less than 12H), then a '.' (period) is printed instead.

You may use the DOS single character output function (INT 21H, AH =02, DL has character to output) or any of the Irvine link library subroutines.

As explained in class, assume the existence of writeint_byte function in the Irvine link library than functions the same as writeint, except that it writes an 8 bit value contained in AL.

```
Dprint  proc    near
        Mov     ax,ds
        Mov     bx,16
        Call    writeint        ;print segment value
        Mov     dl,':'
        Mov     ah,02
        Int     21h             ;print ':'
        Mov     ax,si
        Mov     bx,16
        Call    writeint        ;print offset value
        Mov     cx,16           ;loop 16 times
        Push    si              ;save SI for later
Lp1:    mov     dl, 20h         ;space char
        Mov     ah,02
        Int     21h             ;print space character
        mov     al,[si]
        call    writeint_byte   ;write a byte out
        inc     si              ;point at next byte
        loop    lp1             ;do 16 times

        mov     dl,20h
        mov     ah,02
        int     21h;            ;print a space
;;loop again, print ascii value
        pop     si              ;point back at start
        mov     cx,16           ;loop 16 times
lp2:    mov     dl,[si]
        cmp     dl,12h          ; check if printable ASCII
        jae     do_print        ;if yes, then print it
        mov     dl,'.'          ;get period
do_print:
        mov     ah,02
        int     21h             ;print the character
        loop    lp2             ;loop 16 times
        ret
```