Homework 7

Q1. (**4 pts**) **Memory devices** with 7 address lines, 8 data lines will be used in a memory system for a computer with the following specifications:

- 27-bit address bus, (where A1-A0 are used for a byte access in 8-data lines)
- 32-bit data bus, (which can access bytes, words, and long words)
- Memory at pages 0 to N

Answer the following questions, and show your work.

Q1-1. How many addressable memory locations are in each memory device?

- 7 Address Lines
- 8 Data Lines

2^7 =128
= 128 Addressable Memory Locations

Q1-2. What is the total number of memory devices required in this memory design?

2^27 = 134,217,728 bytes
134,217,728 bytes = 128 MB
134,217,728/128 = 1,048,576 Memory

= 1,048,576 Memory Devices

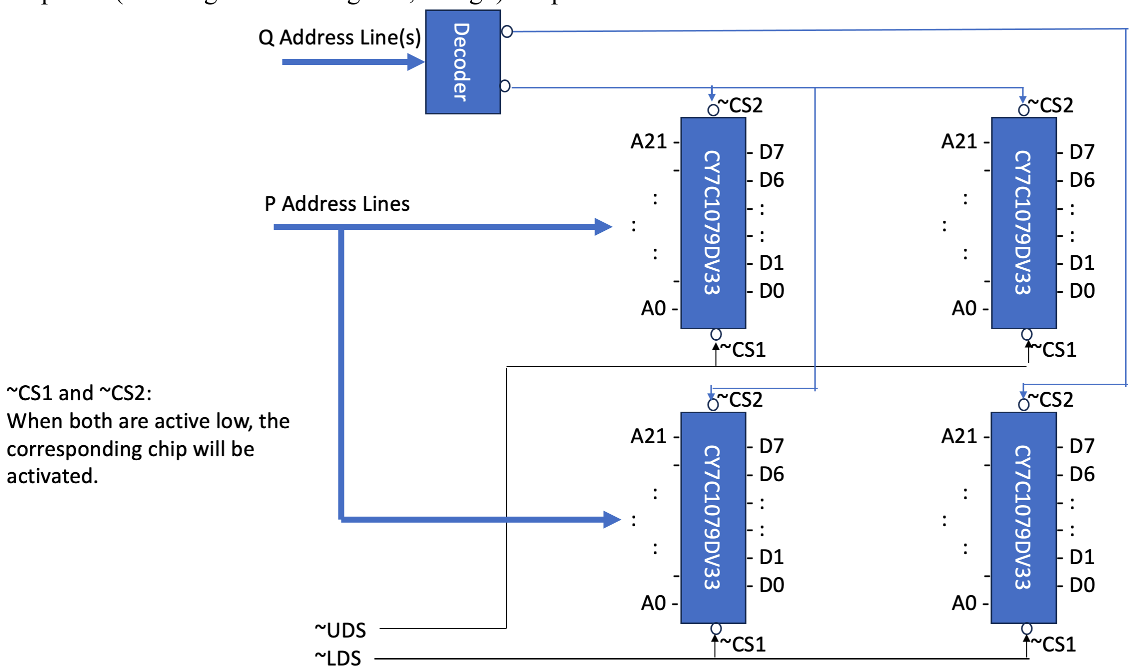
Q2. (6pts) Memory System

Motorola 68K has 16 data lines: D15 – D0; 23 address lines: A23 – A1; and ~UDS and ~LDS data strobe pins. Each of ~UDS and ~LDS is low-active to choose D15 – D8 and D7 – D0 respectively. For example, to access 1 byte at 0x000008, 68K sets A23 – A4 in 0, whereas A3 = 1, A2 = 0, A1 = 0, ~UDS = 1, and ~LDS = 0. Assume that we want to connect 68K to a memory system M that is built with a number of CY7C1079DV33 memory chips, each with 4M x 8-bit wide data. Answer the following four questions.

Q2-1. How many bytes of memory should the memory system M have (The capacity of the memory system in bytes)? 1.5ps

Your answer: = 2^23 = 8,388,608 = 8,388,608 = 8MB **M Memory system should have 8MB total memory**

Q2-2. How many address lines does this memory chip, (CY7C1079DV33) has? In the diagram below, you'll compute P (The diagram has a big hint, though). 1.5pts



Your answer:

4M = 2^22
= 22 Address Lines
P = 22

Q2-3. How many selection bits of each pair of memory chips must be used? In other words, how many upper address lines of 68K should be used for choosing a group of 2 memory chips? In the above diagram, you'll compute Q. (The diagram has a big hint, though). 1.5pts

Your answer
23-bit address
16-but data
4M x 8-bit (P=22)

= 1 Address Line

Q2-4. How many memory chips in total is needed to construct this memory system M? 1.5ps

Your answer

= 8M / 4M
= 2 Chips
= 2 Memory Chips

Q3. (6 pts) Memory-Mapped IO.
Consider two-data transfer cases: 1) by CPU and 2) by DMAC.

Case 1) CPU-initiated data transfer:

Reading data needs:	LDR R0, =src	; 1 cycle
	LDR, R1, [R0]	; 2 cycles
	3 CPU cycles in total	
Writing data needs:	LDR R0, =dst	; 1 cycle
	STR, R1, [R0]	; 2 cycles
	3 CPU cycles in total	

Case 2) DMA-initiated data transfer:
DMAC set-up
CPU needs to write to DMAC's four registers: (1) DMA channel 30 (memory-to-memory transfer)'s source address end pointer, (2) destination address end pointer, and (3) channel 30's control register to issue a transfer request.

```
MOV R1, #imm_src_addr      ; 1 cycle (parameters definition)
LDR R0, =ch30_src          ; 1 cycle
STR, R1, [R0]              ; 2 cycles
MOV R1, #imm_dst_addr      ; 1 cycle (parameters definition)
LDR R0, =ch30_dst          ; 1 cycle
STR, R1, [R0]              ; 2 cycles
MOV R1, #imm_cntrl_data    ; 1 cycle (parameters definition)
LDR R0, =ch30_cntrl        ; 1 cycle
STR, R1, [R0]              ; 2 cycles
```

12 CPU cycles in total

DMAC needs 5 cycles for a 32-bit word transfer from one to another memory.
Now, assume that <u>CPU handles floating-point registers, corresponding to 18 words</u>. Upon receiving an IRQ from DMAC, CPU now needs:

- 3 cycles to switch its CPU mode
- 9 cycles to save regular registers
- <u>18 cycles to save floating-point registers</u>

Given the above two scenarios, at least how many words, (32-bit data) should be transferred if CPU takes advantage of DMAC.

$= 12 + 5N + 30$
$= 5N + 42$

$6N >= 5N + 42$
$6N - 5N >= 42$
$\mathbf{N >= 42}$

Q4. (6pts) Timer Interrupts

The following C program defines sig_handler( ) (lines 2-4) that is invoked upon receiving a SysTick interrupt and that changes alarmed from 1 to 2. The main( ) function (lines 6-16) initializes alarmed to 1 (line 8), schedules sig_hanlder( ) to be invoked upon a SysTick interrupt (line 9), and starts SysTick to count down for 10 seconds (line 10). The main( ) function falls into a while( ) loop (lines 11-14), jumps sig_handler( ) upon receiving a SysTick interrupt, and gets out of the while( ) loop as alarmed eventually becomes 2.

```
 1:     int* alarmed
 2:     void sig_handler( int signum ) {
 3:             *alarmed = 2;
 4:     }
 5:
 6:     int main( ) {
 7:             alarmed = (int *)_malloc( 4 );
 8:             *alarmed = 1;
 9:             _signal( SIG_ALRM, sig_handler );
10:             _alarm( 10 );
11:             while ( *alarmed != 2 ) {
12:                     void* mem9 = _malloc( 4 );
13:                     _free( mem9 );
14:             }
15:     return 0;
16:     }
```

The following Thumr-2 code shows an interrupt handler upon receiving a SysTick interrupt. You don't have to change it at all. Basically this handler invokes the timer_update( ) routine. Your task is to implement timer_update( ).

```
SysTick_Handler\
        PROC
        EXPORT  SysTick_Handler         [WEAK]
        IMPORT  _timer_update
        STMFD   sp!, {lr}               ; save SysTick_Handler's LR
        LDR     R11, =_timer_update
        BLX     R11                     ; invoke timer_update( )

        MRS     R1, PSP                 ;
        STR     R0, [R1]                ; save a return value in PSP
        LDMFD   sp!, {pc}               ; go back to a user program
        ENDP
```

The timer_update( ) function uses the following four parameters.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Timer update
STCTRL          EQU     0xE000E010      ; SysTick Control and Status Register
STCTRL_STOP     EQU     0x00000004      ; Bit2 (CLK_SRC)=1, Bit1 (INT_EN)=0, Bit0 (ENABLE)=0
SECOND_LEFT     EQU     0x20007B80      ; Seconds left for alarm( )
USR_HANDLER     EQU     0x20007B84      ; Address of a user-given signal handler function
```

The timer_update( ) function reads the value of the SECOND_LEFT address, decrements the value by 1 (second), checks the value, branches to _timer_update_done if the value hasn't reached 0, otherwise it needs to stop the timer and to invoke a user function whose address is maintained in the USR_HANDLER address. To stop the timer, write STCTRL_STOP to the address of STCTRL. (Don't forget to save back a decremented value into SECOND_LEFT.)

```
; void timer_update( )
        EXPORT          _timer_update
_timer_update

        LDR  R0, = timeLeft
        LDR  R0, [R0]
        SUB  R0,R0,#1

        LDR  R1  = timeLeft
```

```
        STR  R0, [R1]
        CMP  R0,#0
        BEQ  timeDone
        MOV  pc, lr

        LDR R1, =sysControl
        LDR R2, =sysStop
        STR R2,[R1]

        LDR R1, =timerHandler
        LDR R2,[R1]
        BLX R2


_timer_update_done
        MOV            pc, lr         ; return to SysTick_Handler
```