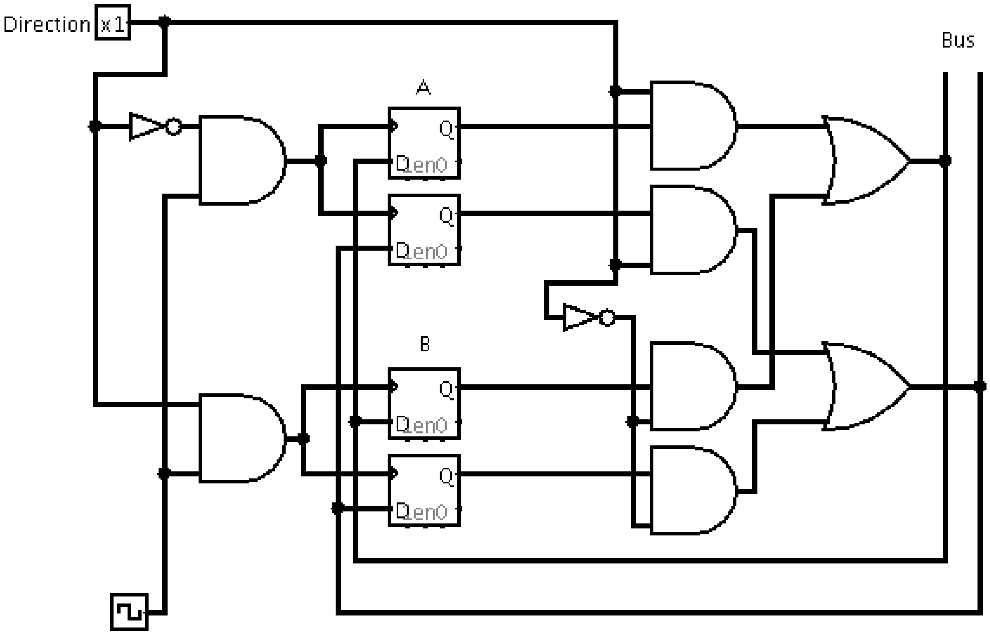
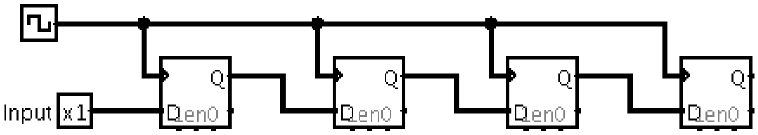
Exam #3 Solutions

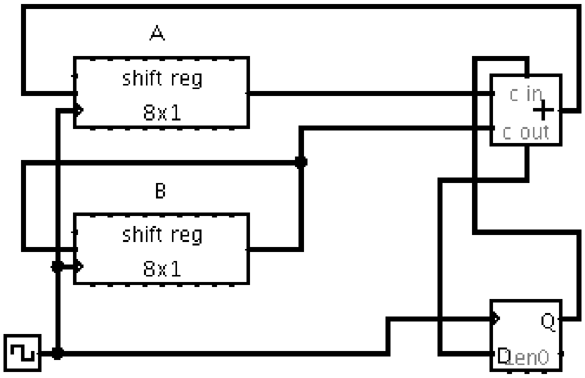
1. (*20 points*) In the following diagram we have 2 two-bit registers A and B constructed from D Flip-Flops and a two-bit bus to the right. There is a clock and a direction selection bit input. Using AND, OR , and NOT gates, construct the circuitry that will transfer register A to register B if the direction bit is 1 and will transfer register B to register A if the direction bit is 0.



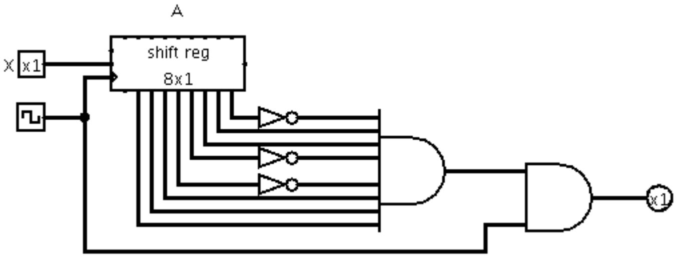
2. (*10 points*) Construct a four-bit shift register using D Flip-Flops.



3. (*20 points*) In the following diagram we have 2 eight-bit shift registers A and B. The register input is on the left and the output is on the right, input and output are both a single bit. Use one full-adder and one D flip-flop to construct a serial adder so that after 8 clock cycles register A contains the sum of registers A and B and register B contains its original value.



4. (*10 points*) In the following diagram we have a single eight-bit shift register A. X is the input stream and the 8 lines coming out of the bottom carry the value of each of the bits in the register. Using AND, OR, and NOT gates, construct a circuit that will recognize the sequence of bits 01100111.



5. (*20 points*) Write a NASM program that will take an informal name (first followed by last) as command-line arguments, and return the person's formal name. Several example runs are below. If the input name is not just first and last then the program should print out an error. You may assume that you have a `functions.asm` file that contains the functions `atoi`, `iprint`, `iprintLF`, `slen`, `sprint`, `sprintLF`, and `quit`, as we did in class.

```
./FormalName Don Spickler
Spickler, Don
./FormalName Jack Frost
Frost, Jack
./FormalName James Earl Jones
Not a valid name.
```

```
%include        'functions.asm'

SECTION .data
errorMsg    db      'Not a valid name.', 0h     ; error message string
sep         db      ', ', 0h                    ; , string

SECTION .text
global  _start

_start:

    pop     ecx             ; the number of arguments
    cmp     ecx, 3          ; check to see if we have 3 arguments.
    jz      continue        ; if zero jump to continue.
    jmp     errorArgs       ; if argument number error, print error message.

continue:
    pop     eax             ; program name
    pop     ebx             ; First name
    pop     eax             ; Last name
    call    sprint          ; Print last name
    mov     eax, sep        ; Load separator
    call    sprint          ; Print separator
    mov     eax, ebx        ; Load first name
    call    sprintLF        ; Print first name

finished:
    call    quit

errorArgs:
    mov     eax, errorMsg   ; load error message
    call    sprintLF        ; Print error message
    call    quit
```

6. (*20 points*) Write a NASM program that will ask the user for a single number n and return the value of $n!$. A couple example runs are below. Recall that $0! = 1$ and $n! = n \cdot (n-1) \cdots 2 \cdot 1$ when $n > 0$. You may assume that you have a `functions.asm` file that contains the functions `atoi`, `iprint`, `iprintLF`, `slen`, `sprint`, `sprintLF`, and `quit`, as we did in class.

```
./prog
Please enter number: 5
5! = 120
./prog
Please enter number: 7
7! = 5040
```

```
%include        'functions.asm'

SECTION .data
msg          db      'Please enter number: ', 0h      ; message string
msgFac       db      '! = ', 0h                       ; output message string

SECTION .bss
sinput:     resb    255     ; reserve a 255 byte space in memory for the users input string

SECTION .text
global  _start

_start:

    mov     eax, msg        ; Load message
    call    sprint          ; Print message

    mov     edx, 255        ; number of bytes to read
    mov     ecx, sinput     ; reserved space to store our input (known as a buffer)
    mov     ebx, 0          ; write to the STDIN file
    mov     eax, 3          ; invoke SYS_READ (kernel opcode 3)
    int     80h

    mov     eax, ecx        ; Move the input string to eax
    call    atoi            ; Convert input string to integer
    mov     esi, eax        ; Store the input count in esi
    mov     ecx, eax        ; Store the input count in ecx for printing later
    mov     eax, 1          ; Start factorial at 1

multLoop:
    cmp     esi, 0h         ; If finished,
    jz      finished        ; jump to the end
    mul     esi             ; multiply next number on factorial
    dec     esi             ; decriment esi
    jmp     multLoop        ; jump to beginning of loop

finished:
    mov     ebx, eax        ; Take sum off the stack
    mov     eax, ecx        ; Move original input to eax
    call    iprint          ; Print output message

    mov     eax, msgFac     ; Load output message
    call    sprint          ; Print output message

    mov     eax, ebx        ; Load output value
    call    iprintLF        ; Print output value

    call    quit
```

7. (*10 points*) Given the following C++ driver program, create the assembly function named `rem` that will take in two input parameters a and b and return the remainder of a/b. As in class, you are using 64-bit assembly in this exercise.

```cpp
#include <iostream>

long rem(long, long) __asm__("rem");

using namespace std;

int main()
{
    long a, b;

    cout << "Input A: ";
    cin >> a;

    cout << "Input B: ";
    cin >> b;

    cout << "Remainder of A/B = " << rem(a, b) << endl;

    return 0;
}
```

```nasm
global  rem
section .text

rem:
    mov rax, rdi
    mov rdx, 0
    div rsi
    mov rax, rdx
    ret
```