CS 245 Fall 2018
Assignment 1A – Ands and Ors
Assignment 1B, 1C – Working with Sprites

For all assignments, be sure to:

- Include file comments at the top, listing your name, the name of the program, and briefly describe what the program does.
- Line up instruction parts in columns: 0=labels; 1tab=instruction mnemonic; 2tabs=operands; 3-4tabs=comments.
- Include pseudo-code comments to the right of your assembly code. **Avoid** comments that tell me what the assembly does, such as: move 25 to register $s3 – I already know this. Keep your comments to the logic.

Program 1A: AND and OR

For the first program, simply perform the following calculation:

$T1 = 0xaa AND 0xf0 OR 0x03
Print your result: "0xaa & 0xf0 | 0x03 = 0x" <N>

Hint: Modify lab1.s to start. You will print your result as a string and an integer. See notes on Assembly logic for print details.

To print a hexadecimal number, set $v0 to 34 and $a0 to the integer to print.
Example code:

```
li      $v0,34          # print_hex(v0=34, a0=solution)
move    $a0,$t1
syscall
```

For more information of what syscall can do, see
http://courses.missouristate.edu/KenVollmar/MARS/Help/SyscallHelp.html.

Using structured code on the side as comments can help you track what you are doing in your code. Below are some comments that you can use to act as compiler to get your program working. You may use as comments the following pseudocode:

```
#                       #  temp = 0xaa AND 0xf0
#                       #  temp = temp OR 0x03
#                       #  print_string("0xaa & 0xf0 | 0x03="
#                       #  print_hex_value(temp)
```

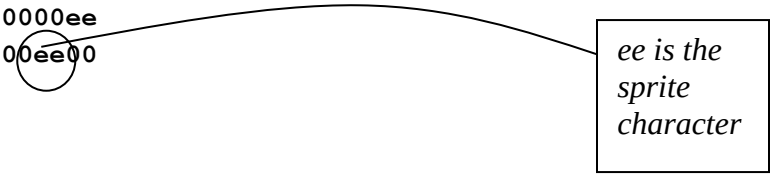
Turn this program in as hwk1.asm via paper and electronic copy to Canvas.

Program 1B: Moving the sprite across one word in memory

Write a MIPS assembly program that when run in MARS at 25 instructions/second will display in the **Memory .data segment** a "sprite" that moves across one word.

Assuming that you chose a "background" of the hex digit 00 and a "sprite" of the hex digit ee, a "time lapse" series of snapshots of the Memory segment display would look like the data below:

Time 1: `0x000000ee`
Time 2: `0x0000ee00`



Time 3: **0x00ee0000**
Time 4: **0xee000000**
Time 5: **0x00ee0000**
Time 6: **0x0000ee00**
Time 7: **0x000000ee**

Or you can do:

Time 1: **0xee000000**
Time 2: **0x00ee0000**
Time 3: **0x0000ee00**
Time 4: **0x000000ee**
Time 5: **0x0000ee00**
Time 6: **0x00ee0000**
Time 7: **0xee000000**

Important: set the MARS slider bar to execute at a slower speed. Omitting this step will run at full speed and no output will be apparent. You may choose the hex digits used for the "sprite" – select two hex digits that have as much contrast as possible for best visibility.

Since assembly code is not self-explanatory (!), your comments are vital for good program clarity. Here are some sample comments you can use. You can act like a compiler and add assembly code to get the program working:

```
#                       #  index = 0;
#                       #  memory.byte[index] = sprite
#                       #  do {
#                       #      index++
#                       #      memory.word = memory.word >> 1 byte
#                       #  } while index < 3
#                       #  do {
#                       #      index—
#                       #      memory.word = memory.word << 1 byte
#                       # } while index > 0
```

The learning goals of this assignment is the use of shifts and loops.

Hint: There is an assembly language shift instruction that makes this program a lot easier. Look for it!

Program 1C: Moving the sprite across multiple words in memory

Write a MIPS assembly program that when run in MARS at 25 instructions/second will display in the **Memory .data segment** a "sprite" that moves across multiple words.

Assuming that you chose a "background" of the hex digits 0 and a "sprite" of the hex digits e, a "time lapse" series of snapshots of the Memory segment display would look like the data below:

Time 1: **0xeeeeeeee 00000000 00000000 00000000**
Time 2: **0x00000000 eeeeeeee 00000000 00000000**

Time 3: `0x00000000 00000000 eeeeeeee 00000000`
Time 4: `0x00000000 00000000 00000000 eeeeeeee`
Time 5: `0x00000000 00000000 eeeeeeee 00000000`
Time 6: `0x00000000 eeeeeeee 00000000 00000000`
Time 7: `0xeeeeeeee 00000000 00000000 00000000`

You may choose the hex digits used for the "sprite" – select hex digits that have as much contrast as possible for best visibility.

The learning goals of this assignment is the use of memory storage manipulation and loops. (Note: Shift will not work here, but load words and store words will.) Manipulate your load and store addresses by adding and subtracting from them to move the sprite across the words. Also use a loop (or two) as part of this assignment.

Turn this program in as hwk1a.asm, hwk1b.asm, and hwk1c.asm via paper and electronic copy to Canvas.

Grading

Each homework assignment is worth 10 points. Be careful to include all comments and format correctly, as directed in the beginning of this file.