Dalhousie University
Department of Electrical and Computer Engineering

ECED 3403 – Computer Architecture

Assignment 2: The X-Makina emulator

1 Objectives

As was shown in *Soul of a New Machine*, before a new computer system is brought to market, hardware and software designers often construct *emulators* to examine how the machine will operate under certain conditions. An emulator "*imitates the function of (another system), as by modifications to hardware or software that allow the imitating system to accept the same data, execute the same programs, and achieve the same results as the imitated system*".[1]

In addition to the emulation of the new computer, support software can be required if the computer has an entirely new architecture. This software can include new compilers, assemblers, linkers, and even operating systems.

Prior to the start of term, the Instruction Set Architecture (ISA) of the X-Makina machine was designed and a two-pass assembler for it was developed. The second assignment requires the design, implementation, and testing of an emulator for X-Makina's ISA. The emulator is to load and execute files containing S-records produced by the assembler. A simple debugger is also required.

The emulator and assembler can run on the machine of your choice. They do not have to run on the same machine since the S-record code is transportable.

2 The Computer

X-Makina's complete ISA is to be emulated, including all CPU registers (R0-R3, R4/LR, R5/SP, R6/PSW, and R7/PC), addressing modes, encoded constants, instructions, and 64kiB of byte-addressable primary memory. External devices and interrupts are also to be supported.[2]

2.1 Instruction cycles

The instruction cycle must be emulated: instructions are fetched from memory, decoded, and executed. The emulator is to recognize each instruction and perform its function. Status changes should be reflected in the PSW's status bits. The CPU's priority and sleep states must be supported by the emulator.

A system clock should be maintained keeping track of the number of instruction cycles a program has used during its execution. The clock can be used when emulating interrupts (see section 2.3.1).

[1] *Emulator*. (n.d.) *American Heritage® Dictionary of the English Language, Fifth Edition*. (2011). Retrieved May 25 2018 from http://www.thefreedictionary.com/emulator

[2] A copy of the X-Makina Instruction Set Architecture can be obtained from http://lh.ece.dal.ca/eced3403/X-MakinaISA.pdf.

2.2 Bus and primary memory

The bus function should take four arguments: the contents of the memory address register, the address of the memory buffer register (to allow data to be returned), the value of the control register (read or write), and an indication whether a byte (1) or a word (0) is to be accessed (by rights, this should be part of the control register); for example:

```
/* To read: */
mar = effective_address;  /* Address to be read */
bus(mar, &mbr, READ, byte_word);
/* mbr has contents of location specified by 'address' */

/* To write: */
mar = effective_address; /* Address to be written */
mbr = data;
bus(mar, &mbr, WRITE, byte_word);
```

Primary memory will require special code (i.e., soft "circuitry") that supports the CPU reading and writing to the memory reserved for devices (also referred to as *port memory* or *device ports*).

2.3 Interrupts and devices

The current version of X-Makina has a 16-entry interrupt vector table stored in high memory, locations 0xFFE0 through 0xFFFE. Each vector is 32-bits long. The least-significant 16-bit word contains the PSW of the interrupt service routine (ISR) and the most-significant word contains the 16-bit address of the ISR:

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nnnn	-	-	-	-	-	-	-	-	Priority			V	0	N	Z	C
nnnn+2	16-bit address of ISR															

For an interrupt to occur, the following rules are applied:

1. A device's priority (specified in the Priority field of its interrupt vector) must be greater than the machine's current priority (i.e., the priority in the PSW). This means that nested interrupts are possible.
2. If two or more devices interrupt simultaneously, the highest priority device will be serviced first.
3. If two or more devices interrupt simultaneously and have the same priority, the device with the lowest vector number will be serviced first.
4. If two or more interrupts are pending after an ISR is completed, rules 1, 2, and 3 are applied to determine which device is serviced first.

The steps involved in saving and restoring the state of the interrupted code is described in section 9 of the X-Makina ISA Summary document. The sleep-state indicator (SLP, bit 3 of the PSW) should be clear to ensure that the ISR does not immediately enter the sleep-state.

The assignment is to allow up to eight devices to be used (each associated with a program-specified priority, see above). The first eight vectors are assigned to the devices. The remaining vectors will be used for CPU faults and program traps. Vector 15 is reserved for CPU resets.

Each device is associated with a one-word port consisting of a control/status register and data register:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data (I/O)								Reserved				OF	DBA	I/O	IE

where:

IE: Interrupt enable (bit 0). Indicates whether this device is enabled for interrupts. Enable by setting the bit. If the bit is not enabled, the device can still be active; however, in order to determine whether a status change has occurred, it is necessary to poll the device. This is a control bit.

I/O: Input/output enable (bit 1). Indicate whether the device is for input (set bit) or output (clear bit). This is a control bit.

DBA: Data or buffer available (bit 2). If the device is programmed to be an input device and this bit is set, it means that data is available for reading. If the device is programmed to be an output device and this bit is clear, it means that the buffer is available for writing. The bit is cleared by the device when an input byte is read and set when the device writes a byte to the buffer. The bit is cleared by the device when an output byte is written and set by the device when the byte is transmitted. This is a status bit.

OF: Overflow (bit 3). This bit is set if the device is enabled for input and a byte is overwritten by a subsequent byte (i.e., it was not read quickly enough) or the device is enabled and the CPU writes bytes to the buffer before the device has time to transmit the byte. This is a status bit.

Reserved: These bits are reserved for future use or device-specific encodings (bits 4 through 7).

Data (I/O): These bits are for input or output, depending on how the device has been defined and subsequently enabled (bits 8 through 15).

The device ports are located in addresses 0x0000 through 0x000E. Each device is associated with a device number, which can be used to access either the device's port or its interrupt vector. For example, device 0 has port 0x0000 and vector 0xFFC0.

Device ports can be accessed as byte-pairs, the even-numbered byte holding the control/status register and the odd-numbered byte being the data register.

An interrupt occurs *after* an instruction has completed execution.

2.3.1 Testing interrupts

Interrupts can occur at random intervals during a program's execution; however, for testing purposes, it makes more sense to control when the interrupt occurs. There are a number of ways in which this can be done; the following is a description of one such method.

Causing an interrupt on an interrupt-enabled input port can be done by creating a file which specifies the time of the interrupt, the port, and the data associated with the interrupt. When the system-clock time is equal-to or greater-than the specified interrupt time, the emulator can trigger the interrupt if allowed (this includes setting the port's DBA bit and writing the data in the port's memory location). By specifying several interrupts at the same time in the file, concurrent interrupts can be tested.

When an interrupt-enabled output port is written to, the time, port, and contents can be written to a file recording any events that change the state of the machine. Since the output is not instantaneous, a port-specific counter (internal to the emulator) can be assigned a value which is decremented each time the system clock is incremented; when this counter reaches zero, the port can interrupt the CPU. Each output device will require its own counter.

3 Support software

3.1 Loader

It will be necessary to include a loader as part of the X-Makina emulator. The loader will take files containing S-records (see the X-Makina Assembler User's Guide) and load the contents of the file into the memory locations specified. The program counter must be initialized to the value in the S9 record. If the S9 record is omitted, the Reset vector is to be used, supplying the PSW (including the startup priority) and the address of the startup subroutine (see section 10 of the X-Makina ISA Summary).

3.2 Debugger

A debugger is also to be part of the emulator. This is to allow the user to load programs, start a program, stop it, and inspect or change CPU registers and the contents of memory locations. The debugger should also allow the programmer to set breakpoints to stop a program's execution.

To catch run-away programs, a control-C signal catcher should be implemented.

4 Marking

The assignment will be marked using the following marking scheme:

Design: The design description must include a brief introduction as to the purpose of the software and describe the algorithms and data structures used.

Total points: 8.

Software: A fully commented, indented, magic-numberless, tidy piece of software that meets the requirements described above and follows the design description.

Total points: 12.

Testing: In addition to any test software supplied as part of the assignment, you will be responsible for developing a minimum of five distinct tests demonstrating that the software operates according to the design description. Some of the tests should exercise the software. The tests must include the name of the test, its purpose or objective, the test configuration, and the test results.

Total points: 5.

The assignment must be submitted on paper. The executable must be submitted as well; how this is to be done will be explained in class.

The emulator must be demonstrated before the software and testing will be marked.

5 Important Dates

Available: 30 May 2018

Design document due: 21 June 2018 (3pm, at start of class).

Demonstrations and submission of program and testing: 5 July 2018 (3pm, at start of class).

Late assignments will be penalized two points per week or fraction thereof.

The last day this assignment will be accepted is 19 July 2018.

6 Miscellaneous

This assignment is to be completed individually.

Do not discard this work when completed, as it will be used in the final assignment.

If you are having *any* difficulty with this assignment or the course, *please* contact Dr. Hughes as soon as possible.

This assignment is worth 25% of your overall course grade.

This is a non-trivial assignment. It should be started as soon as it is made available.

Assignments that are found to be copies of work done by other students will result in an immediate dismissal from this course.