| Friday | **Assignment-4** | Due on: |
|---|---|---|
| April 9, 2021 | **EE 309: MicroProcessors** | April 18, 2021 |
| | Spring Semester 2021 | Before 23:50 |

**Q–1** We have developed software for scanning a keyboard using FSMs earlier with an 8051. We now want to do it using a MIPS processor. In this case, we want to write the operation of FSM as a callable function, which will be called at intervals of 50 ms. (Timer operation and interrupt handling is not to be implemented here).
Assume that there is an external chip, which responds to a memory address 0x40000000, with the same functionality as 8051 port 0. (The external chip connects to MIPS like a memory and on the output side, has 8 open collector outputs like 8051 port0, connected the same way to the keyboard with external pullups). Thus, writes and reads to/from Port0 of 8051 are equivalent to unsigned byte store and unsigned byte load to/from the memory address 0x4000000.

Write the equivalent keyscan FSM function with the same state diagram, using MIPS assembly language.

Your code should not be a literal translation of the 8051, but should make full use of resources and conventions used for MIPS.

Your submission should have an a '.s' file with code and a pdf file for text answers.

Assignment Ends

```
Hints:

You are allowed to use pseudo-instruction.
(Without these, it might be too laborious
particularly for loading addresses to
initialize pointer.

Psudo-instruction la loads the address of
some label into a register.

-> Labels are case sensitive in qtspim.

-> Qtspim emulates a MIPS processor using
Little Endian convention for multi-byte data!

-> Notice that the port address has been
changed to 0x40000000 to avoid clashes with
qtspim memory allocation.

My data segment is as follows -- you need not
use the same arrangement, of course.
```

```
Labels AnyKey etc. must match the ones used
in your program exactly.

.data
TestTab: .word AnyKey, TheKey
ActTab:  .word DoNothing, FindKey, ReportKey
Cur_St: .byte 0x00
Key_Code: .byte 0x7E
Key_Buffer: .space 16
In_Dest: .byte 0
Out_From: .byte 0
Periph: .space 4
# State Diagram Data
Test_No: .byte ...  ...  ...  ...
Yes_Act: .byte ...  ...  ...  ...
No_Act: .byte ...  ...  ...  ...
Yes_Next: .byte ...  ...  ...  ...
No_Next: .byte ...  ...  ...  ...
.text
.globl main
main:

Please put a nop after every branch and call instr.
This has to do with the pipelined design of MIPS.
You may get away without doing so during emulation,
but a real pipelined processor may run into
problems if you don't. (see the lecture on Apr. 12).

You should test your program by single stepping.
You can put breakpoints by right clicking on
instructions. You can modify register/memory
contents by right clicking on them.

Run through all state transistions  by modifying
the byte read from PortAddr to what you would
expect from a pressed/not pressed key.
This is what you will demonstrate to TAs when
the assignment is checked.



Links have been put on moodle site for excellent
tutorials on qtspim and MIPS assembly programming.
```