

3.1 [2 points] Typescript for compilation

Turn in a typescript showing compilation of your code using a modified Makefile (same as for cooperative except the file names are changed to the preemptive version). You should use the following two commands (Note: \$ is the prompt displayed by the shell and is not part of the command that you type.) The first one deletes all the compiled files so it forces a rebuild if you have compiled before. The second one compiles it.

```
$ make clean
$ make
```

It should show actual compilation, warning, or error messages. Note that not all warnings are errors. The compiler should generate several `testpreempt.*` files with different extensions:

- the `.hex` file can be opened directly in EdSim51
- the `.map` file shows the mapping of the symbols to their addresses after linking

```
typescript.txt
Script started on Mon Nov  9 18:59:20 2020
[1m[7m[27m[1m[0m

]2;mittyu@Mittyus-MacBook-Pro: ~/Desktop/BD3/OS/myProject/107034024-
ppc2]1;..07034024-ppc2
[0m[27m[24m[J[01;32m→ [36m107034024-ppc2[00m [01;34mgit:([31mmaster[34m)
[33m×[00m [K[?1h=[?2004hmmake clean n[?1l>[?2004l

]2;make clean]1;makerm *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym
rm: *.ihx: No such file or directory
rm: *.lnk: No such file or directory
make: *** [clean] Error 1
[1m[7m[27m[1m[0m

]2;mittyu@Mittyus-MacBook-Pro: ~/Desktop/BD3/OS/myProject/107034024-
ppc2]1;..07034024-ppc2
[0m[27m[24m[J[01;32m→ [36m107034024-ppc2[00m [01;34mgit:([31mmaster[34m)
[33m×[00m [K[?1h=[?2004hmmake[?1l>[?2004l

]2;make]1;makesdcc -c testpreempt.c
testpreempt.c:54: warning 158: overflow in implicit constant conversion
sdcc -c preemptive.c
preemptive.c:208: warning 85: in function ThreadCreate unreferenced function
argument : 'fp'
sdcc -o testpreempt.hex testpreempt.rel preemptive.rel
[1m[7m[27m[1m[0m

]2;mittyu@Mittyus-MacBook-Pro: ~/Desktop/BD3/OS/myProject/107034024-
ppc2]1;..07034024-ppc2
[0m[27m[24m[J[01;32m→ [36m107034024-ppc2[00m [01;34mgit:([31mmaster[34m)
[33m×[00m [K[?1h=[?2004heexit[?1l>[?2004l

]2;exit]1;exit
Script done on Mon Nov  9 18:59:37 2020
```

3.2 [18 points] Screenshots and explanation

Look up the addresses for your symbols (i.e., functions, variables, etc) in the file `testpreempt.map`. Set one or more breakpoints in EdSim51's assembly code window after you have assembled it.

- Take one screenshot before each `ThreadCreate` call. Explain how the stack changes.

The screenshot displays the EdSim51 software interface, which is used for simulating the 8051 microcontroller. The interface is divided into several sections:

- System Clock (MHz):** Set to 11.0592. A dropdown menu shows 1000, and an "Update Freq." button is present.
- Registers:** A table showing the status of various registers (R0-R7, ACC, PSW, IP, IE, PCON, DPH, DPL, SP). The PC register is highlighted with the value 8051.
- Data Memory:** A table showing memory addresses (0000-00FF) and their corresponding values. The PC register is highlighted with the value 8051.
- Assembly Code Window:** Displays the assembly code for the program. The code includes instructions like `RET`, `LJMP 020AH`, `RETI`, `MOV 89H, #00H`, `MOV 0A8H, #82H`, `SETB 8CH`, `MOV 35H, #00H`, `MOV 3AH, #01H`, `MOV 3BH, #02H`, `MOV 3CH, #04H`, `MOV 3DH, #08H`, `MOV DPTR, #0066H`, `LCALL 00BAH`, `MOV 34H, 82H`, `MOV A, 34H`, `ADD A, #30H`, `MOV R1, A`, `MOV 81H, @R1`, `POP 0D0H`, `POP 83H`, `POP 82H`, `POP 0F0H`, `POP 0E0H`, `RET`, and `MOV A, #0FH`. A breakpoint is set at address 00BA.
- Hardware Components:** A panel at the bottom shows various hardware components and their status:
 - DI / LD:** A switch labeled "DI / LD" is shown.
 - AND Gate Disabled:** A checkbox labeled "AND Gate Disabled" is checked.
 - Key Bounce Disabled:** A checkbox labeled "Key Bounce Disabled" is checked.
 - Standard:** A dropdown menu showing "Standard".
 - 0.0 V output:** A label for the output voltage.
 - Scope DAC:** A label for the scope DAC.
 - 8-bit UART @ 4800 Baud:** A label for the UART configuration.
 - Rx Reset:** A button labeled "Rx Reset".
 - Tx Send:** A button labeled "Tx Send".
 - ADC:** A label for the ADC component.
 - Motor Enabled:** A checkbox labeled "Motor Enabled" is checked.

- Take one screenshot when the Producer is running. How do you know?

The screenshot displays a microcontroller development environment with the following components:

- System Configuration:** System Clock (MHz) 11.0592, Update Freq. 1000.
- Registers:** R0-R7, B, ACC, PSW, IP, IE, PCON, DPH, DPL, SP. The PC register is highlighted at 8051.
- Data Memory:** A table showing memory addresses 00 to 70 and their corresponding values.
- Assembly Code:** A list of instructions including MOV, CJNE, RET, JBC, MOV, INC, SJMP, ADD, RLC, DJNZ, and MOV.
- Hardware Simulation:**
 - Keypad:** A 4x4 grid with labels DI, LD, and a numeric keypad.
 - UART:** Configuration for 8-bit UART @ 4800 Baud, No Parity. Includes Rx and Tx fields and buttons for Rx Reset and Tx Send.
 - ADC:** Configuration for 0.0 V input, MAX/MIN range, and a Motor Enabled button.
 - Display:** A 7-segment display showing the value 8888.
 - Scope:** A scope window showing a signal waveform.

- Take one screenshot when the Consumer is running. How do you know?

The screenshot displays the Proteus ISIS simulation environment for the 'Consumer' project. The top section shows the Register File with registers R0-R7, ACC, PSW, IE, PCON, DPH, DPL, and SP. The Data Memory window shows a memory dump with addresses 00 to 70. The Instruction Register shows the current instruction: `MOV A, #0FH`. The bottom section shows the hardware components: a keypad, a 4-digit display showing '8888', an ADC with a value of 11111111, and a motor control section with 'Motor Enabled'.

- How can you tell that the interrupt is triggering on a regular basis?