

3. Typescript and screenshots

3.1 Typescript for compilation

```
PS C:\Users\peggy\Downloads\OS_cp2> make clean
del *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym *.asm *.lk
PS C:\Users\peggy\Downloads\OS_cp2> make
sdcc -c testpreempt.c
testpreempt.c:79: warning 158: overflow in implicit constant conversion
sdcc -c preemptive.c
sdcc -o testpreempt.hex testpreempt.rel preemptive.rel
```

3.2 Screenshots and explanation

Memory location of functions:

| | Value | Global | Global Defined In Module |
|-----|-------|------------------------------|--------------------------|
| 286 | | | |
| 287 | | | |
| 288 | C: | 00000014 _Producer | testpreempt |
| 289 | C: | 00000040 _Consumer | testpreempt |
| 290 | C: | 00000065 _main | testpreempt |
| 291 | C: | 00000074 _sdcc_ginit_startup | testpreempt |
| 292 | C: | 00000078 _mcs51_genRAMCLEAR | testpreempt |
| 293 | C: | 00000079 _mcs51_genXINIT | testpreempt |
| 294 | C: | 0000007A _mcs51_genXRAMCLEAR | testpreempt |
| 295 | C: | 0000007B _timer0_ISR | testpreempt |
| 296 | C: | 0000007F _bootstrap | preemptive |
| 297 | C: | 000000A8 _ThreadCreate | preemptive |
| 298 | C: | 00000128 _ThreadYield | preemptive |
| 299 | C: | 000001B9 _ThreadExit | preemptive |
| 300 | C: | 000001A6 _myTimerHandler | preemptive |

Memory location of variables:

| | | | |
|----|-------|------------|--------------------|
| 16 | _data | _at (0x30) | char savedSP[4]; |
| 17 | _data | _at (0x34) | char threadBitmap; |
| 18 | _data | _at (0x35) | char threadId; |
| 19 | _data | _at (0x36) | char threadId_new; |
| 20 | _data | _at (0x37) | char threadCount; |
| 21 | _data | _at (0x22) | char tempSP; |

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

ThreadCreate for main/Consumer:

The screenshot displays the SIM51 IDE interface. The main window shows the assembly code for the ThreadCreate function. The code starts with a JBC instruction at address 0060, followed by a SJMP instruction at 0063. The code then moves values into registers R3, R4, and R5, and sets up the stack pointer (SP) to 0014H. The code then calls the LCALL instruction at 006E, which is the ThreadCreate function. The code then returns (RET) at 0077 and 0079, and finally jumps to the main function at 007A. The memory dump on the right shows the stack contents, with the top of the stack at address 0014H. The stack contains the return address 007A, followed by the arguments 0000, 0000, 0000, and 0000. The stack pointer (SP) is currently at 0014H.

ThreadCreate for Producer:

The screenshot displays the Proteus IDE interface for an 8051 microcontroller simulation. The CPU window shows the program counter (PC) at 0x0090 and the instruction pointer (IP) at 0x0000. The data memory window shows the value 0x00 at address 0x00. The I/O window shows the output of the DAC as 0.0V. The UART window shows the input and output data.

- Take one screenshot when the Producer is running. How do you know?
Current thread ID (stored at 0x35) is 1, which is the thread ID of Producer.

The screenshot displays the Proteus IDE interface for an 8051 microcontroller simulation. The CPU window shows the program counter (PC) at 0x001C and the instruction pointer (IP) at 0x0001. The data memory window shows the value 0x01 at address 0x00. The I/O window shows the output of the DAC as 0.0V. The UART window shows the input and output data.

- Take one screenshot when the Consumer is running. How do you know? Current thread ID (stored at 0x35) is 0, which is the thread of Consumer.

System Clock (MHz) 11.0592 Update Freq. 100

Time: 38ms 556us - Instructions: 22316

Assembly Code:

```

001C| JUMP 0F9H
001E| SETB 00H
0020| JBC 0AFH,02H
0023| CLR 00H
0025| MOV 30H,21H
0028| MOV 39H,#01H
002B| MOV A,#5AH
002D| CJNE A,21H,05H
0030| MOV 21H,#41H
0033| Sjmp 05H
0035| MOV A,21H
0037| INC A
0038| MOV 21H,A
003A| MOV C,00H
003C| MOV 0AFH,C
003E| JUMP 0D7H
0040| ORL 89H,#20H
0043| MOV 8DH,#0FAH
0046| MOV 98H,#50H
0049| SETB 8EH
004B| MOV A,39H

```

Memory Window (0x00 to 0x00):

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 30 | 31 | 00 | 00 | 03 | 00 | 00 | 02 | 31 | 32 | 80 | 00 | 03 | 00 | 04 |
| 10 | 00 | 30 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 20 | 03 | 42 | 42 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 30 | 46 | 56 | 00 | 00 | 03 | 01 | 02 | 41 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 40 | 4B | 00 | 00 | 00 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 50 | 1C | 00 | 01 | 00 | 00 | 09 | 30 | 31 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 60 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 70 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

Hardware Window:

- DI: 1, LD: 1
- AND Gate Disabled
- Key Bounce Disabled
- Standard
- U: No Parity, 8-bit UART @ 4800 Baud
- Rx: A, Tx: Send
- ADC: 0.0 V, 11111111
- Motor Enabled

- How can you tell that the interrupt is triggering on a regular basis? Since myTimer0Handler is located at 0x01A6, we set a breakpoint at 01A6 of assembly code.

System Clock (MHz) 11.0592 Update Freq. 1000

Time: 497ms 793us - Instructions: 263276

Assembly Code:

```

019B| POP 82H
019D| POP 0F0H
019F| POP 0E0H
01A1| POP 0D0H
01A3| MOV 0AFH,C
01A5| RET
01A6| CLR 0AFH
01A8| PUSH 0E0H
01AA| PUSH 0F0H
01AC| PUSH 82H
01AE| PUSH 83H
01B0| PUSH 0D0H
01B2| MOV A,35H
01B4| ADD A,#30H
01B6| MOV R0,A
01B7| MOV R0,81H
01B9| PUSH 00H
01BB| PUSH 01H
01BD| MOV A,#0FDH
01BF| ADD A,35H

```

Memory Window (0x00 to 0x00):

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 30 | 31 | 00 | 00 | 03 | 00 | 00 | 02 | 31 | 32 | 80 | 00 | 03 | 00 | 04 |
| 10 | 00 | 30 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 20 | 03 | 42 | 42 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 30 | 46 | 56 | 00 | 00 | 03 | 01 | 01 | 02 | 41 | 01 | 00 | 00 | 00 | 00 | 00 |
| 40 | 4B | 00 | 00 | 00 | 01 | 00 | 80 | 30 | 31 | 00 | 00 | 00 | 00 | 00 | 00 |
| 50 | 17 | 00 | 01 | 00 | 00 | 09 | 30 | 31 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 60 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 70 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

Hardware Window:

- DI: 1, LD: 1
- AND Gate Disabled
- Key Bounce Disabled
- Standard
- U: No Parity, 8-bit UART @ 4800 Baud
- Rx: ABCDEFGHIJKLMNOPQRSTUVWXYZ, Tx: Send
- ADC: 0.0 V, 11111111
- Motor Enabled