



EHB326E – INTRODUCTION TO EMBEDDED SYSTEMS HW4

Instructor: Prof. Dr. Müştak Erhan Yalçın
Assistant: Alp Eren Kıyak

Kamil Eren Ezen
040210021

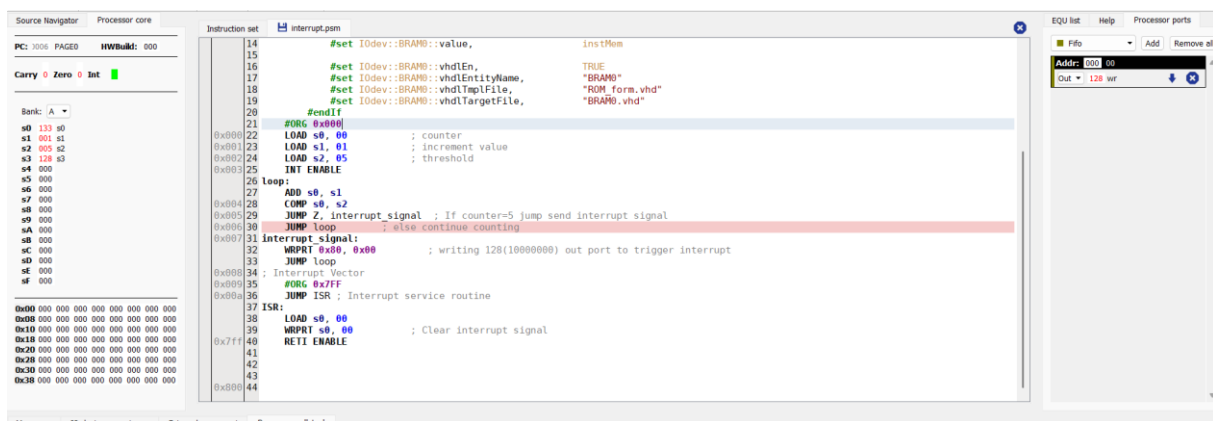
Call/Return Stack:

The call/return stack is feature that enables efficient function calls and handling of interrupts. It operates as a Last In, First Out (LIFO) structure, storing return addresses when subroutines or interrupts are called. The Picoblaze stack has a finite depth of 31 levels. Exceeding this depth results in stack overflow, potentially causing loss of return addresses or incorrect program behavior.

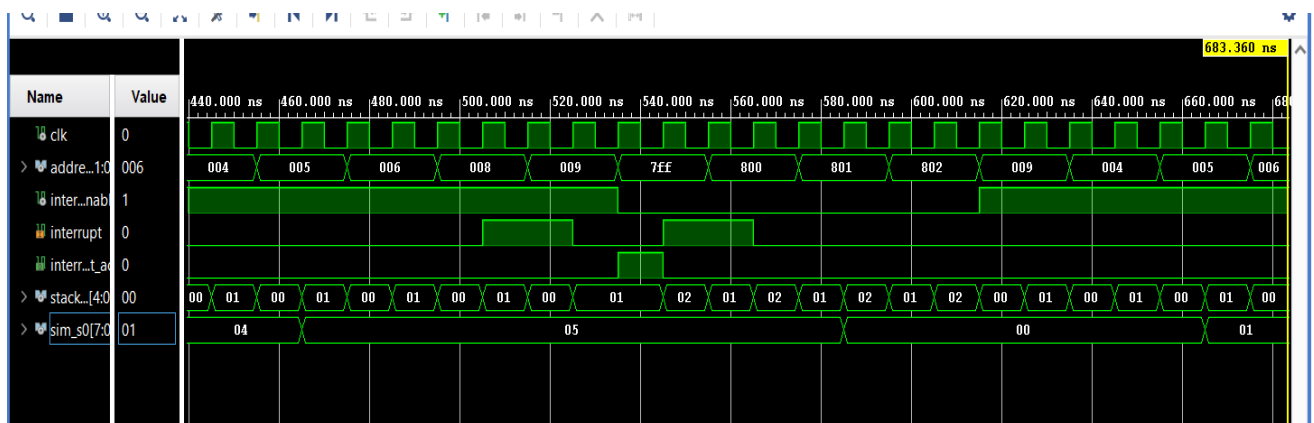
Interrupt:

Interrupts allow the Picoblaze processor to temporarily pause its current task to handle critical or time-sensitive events, such as external signals or internal conditions.

Interrupt Implementation



For the assembly part I wrote a simple counter code including interrupt. It starts counting then being interrupted after reaching 5.



Address instruction goes to interrupt address while interrupted. Then continues from where it left.

The screenshot displays the Keil uVision IDE interface with the following components:

- Source Navigator:** Shows the project structure with files like `PC: J003 PAGE0` and `HWBuild: 000`.
- Processors core:** Displays the processor configuration, including `Bank: A` and various registers like `s0`, `s1`, `s2`, etc.
- Assembly View:** Shows the assembly code for the `nestedInterrupt.psm` file. The code includes comments and instructions for setting up a nested interrupt, such as `#set I0dev::BRAM0::value, instMem` and `WRPRT 0x80, 00`.

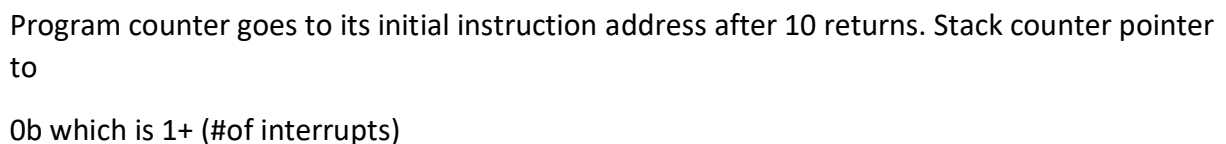
The assembly code is as follows:

```

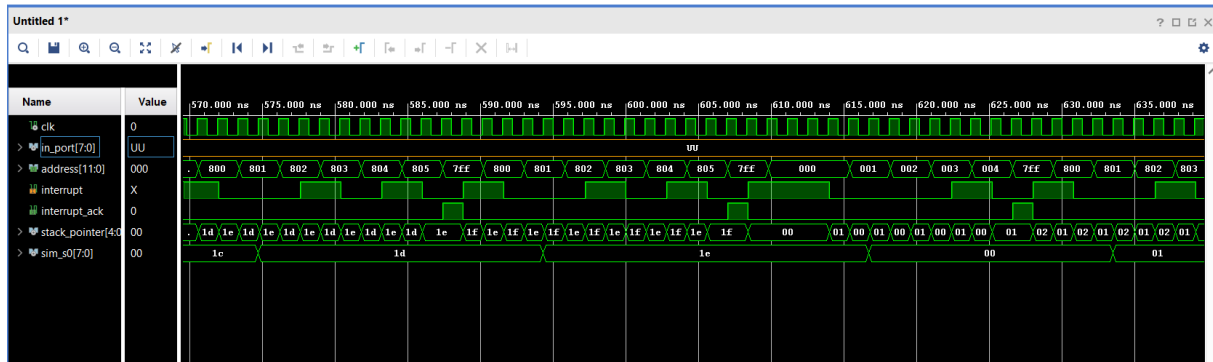
13
14      #set I0dev::BRAM0::value,      instMem
15
16      #set I0dev::BRAM0::vhd1En,      TRUE
17      #set I0dev::BRAM0::vhd1EntityName, "BRAM0"
18      #set I0dev::BRAM0::vhd1TplFile,  "ROM_form.vhd"
19      #set I0dev::BRAM0::vhd1TargetFile, "BRAM0.vhd"
20
21      #endif
22      #ORG 0x000
23      LOAD s0, 00      ; Interrupt counter
24      LOAD s1, 10      ; number of interrupts
25      INT ENABLE
26
27      25 loop:
28      WRPRT 0x80, 00      ; Write to I/O port 0x80 to trigger interrupt
29      JUMP loop
30
31      28 ; Interrupt Vector
32      #ORG 0x7FF
33      JUMP ISR ; Interrupt Service Routine
34
35      31 ISR:
36      ADD s0, 01
37      COMP s0, s1
38      JUMP Z, EXIT_nested ; If max nested interrupt level reached exit
39      INT ENABLE          ; Re-enable interrupts for nesting
40      WRPRT 0x80, 00
41      RETI ENABLE
42
43      38 EXIT_nested:
44      ; Reset and clear interrupt once nesting is complete
45
46      0x806 40      LOAD s0, 00
47      0x807 41      WRPRT s0, 00
48      0x808 42      RETI ENABLE
49      43

```

Here are the simulation results:



Simulation results for 32 interrupts. (Only changed s1 value to 32.)



After interrupt 31 the program overflows before reaching interrupt 32. Because stack pointer reaches its maximum $1f = 1 + 31$. Which is why 32 nested interrupts are not possible in PicoBlaze. Program counter does not return to its initial place.