cpulator.01xz.net/?sys=rv32-de1soc

Stopped | Step Into F2 | Step Over Ctrl-F2 | Step Out Shift-F2 | Continue F3 | Stop F4 | Restart Ctrl-R | Reload Ctrl-Shift-L | File | Help

Registers

Refresh

pc	00000036
zero x0	00000000
ra x1	00000026
sp x2	00020000
gp x3	00000000
tp x4	00000000
t0 x5	ff200000
t1 x6	ff200040
t2 x7	000fd601
s0 x8	ff200040
s1 x9	ff200000
a0 x10	0000000a
a1 x11	00000064
a2 x12	00000000
a3 x13	00000000
a4 x14	00000000
a5 x15	00000000
a6 x16	00000000
a7 x17	00000000
s2 x18	ff200050
s3 x19	00000000
s4 x20	00000079
s5 x21	ff200050
s6 x22	00000000
s7 x23	00000000
s8 x24	00000000
s9 x25	00000000
s10 x26	00000000
s11 x27	00000000
t3 x28	00000041
t4 x29	04444444
t5 x30	00000000

Registers | Call stack | Trace | Breakpoints | Watchpoints | Symbols | Counters

Settings

Number Display Optio

Size: Word

Format: Hexadecimal

Memory words per row: 4

Editor (Ctrl-E)

Compile and Load (F5) | Language: RV32 | part1.s

```
1  # Program that displays SW switch settings on LEDR. When a KEY is pressed,
2  # an interrupt is generated and the handler displays SW on HEX0
3  .equ LEDR_BASE, 0xff200000
4  .equ HEX3_HEX0_BASE, 0xff200020
5  .equ SW_BASE, 0xff200040
6  .equ KEY_BASE, 0xff200050
7  .equ KEY_INT, 0x40000                  # KEY port is IRQ 18
8
9  .global _start
10
11 _start:     li      sp, 0x20000        # set up the stack
12             la      t0, handler
13             csrw    mtvec, t0          # set trap address
14             li      t0, KEY_INT        # KEY port interrupt
15             csrs    mie, t0            # enable KEY interrupts
16             csrsi   mstatus, 0b1000    # enable Nios V interrupts
17
18             jal     setup_KEYs
19
20             li      t0, LEDR_BASE      # pointer to LEDR port
21             li      t1, SW_BASE        # pointer to SW port
22 loop:       lw      t3, (t1)           # read from SW
23             sw      t3, (t0)           # write to LEDR
24             j       loop
25
26 # Trap handler
27 handler:    addi    sp, sp, -12        # save regs that will be modified
28             sw      t2, 8(sp)
29             sw      t1, 4(sp)
30             sw      t0, (sp)
31
32             # check for cause of exception
33             csrr    t0, mcause         # read mcause register
34             li      t1, 0x80000012     # 0x8... checks the interrupt bit
35             xor     t1, t1, t0         # 0x12 (18) is IRQ # for KEY port
36 stay:       bnez    t1, stay           # unexpected cause of trap
37
38             li      t0, KEY_BASE       # clear the interrupt
39             lw      t1, 12(t0)
40             sw      t1, 12(t0)
41
42             li      t0, SW_BASE        # pointer to SW port
```

Editor (Ctrl-E) | Disassembly (Ctrl-D) | Memory (Ctrl-M)

Devices

LEDs ff200000

Switches ff200040

9 8 7 6 5 4 3 2 1 0 All

Push buttons IRQ 18 ff200050

3 2 1 0 All

Seven-segment displays ff200020

JTAG UART IRQ 24 ff201000

```
Contents of s1: 0xFF200050
Contents of s1: 4280287312
Contents of s1: 4280287312
Contents of s1: 4280287312
Contents of s1: 0xFF200050

JTAG UART example code
> jkjkj
jkjk
```

Read FIFO: 0 Write FIFO: 0

RISC-V Machine Timer IRQ 7 ff202100

8524575435 >= mtimecmp: 1000000

Interval Timer IRQ 16 ff202000

6249999 Once Stop TO=0

Interval Timer IRQ 17 ff202020

6249999 Once Stop TO=0

VGA pixel buffer 08000000

Messages

Clear

```
Compiling...
Code and data loaded from ELF executable into memory. Total size is 172 bytes.
Assemble: riscv32-as -march=rv32imf_zicsr_zicbom --gdwarf2 -o work/asmu2LzXt.s.o work/asmu2LzXt.s
Link: riscv32-ld  --script build_mips.ld -e main -u main -o work/asmu2LzXt.s.elf work/asmu2LzXt.s.o
Compile succeeded.
```