Lab Assignment 1:

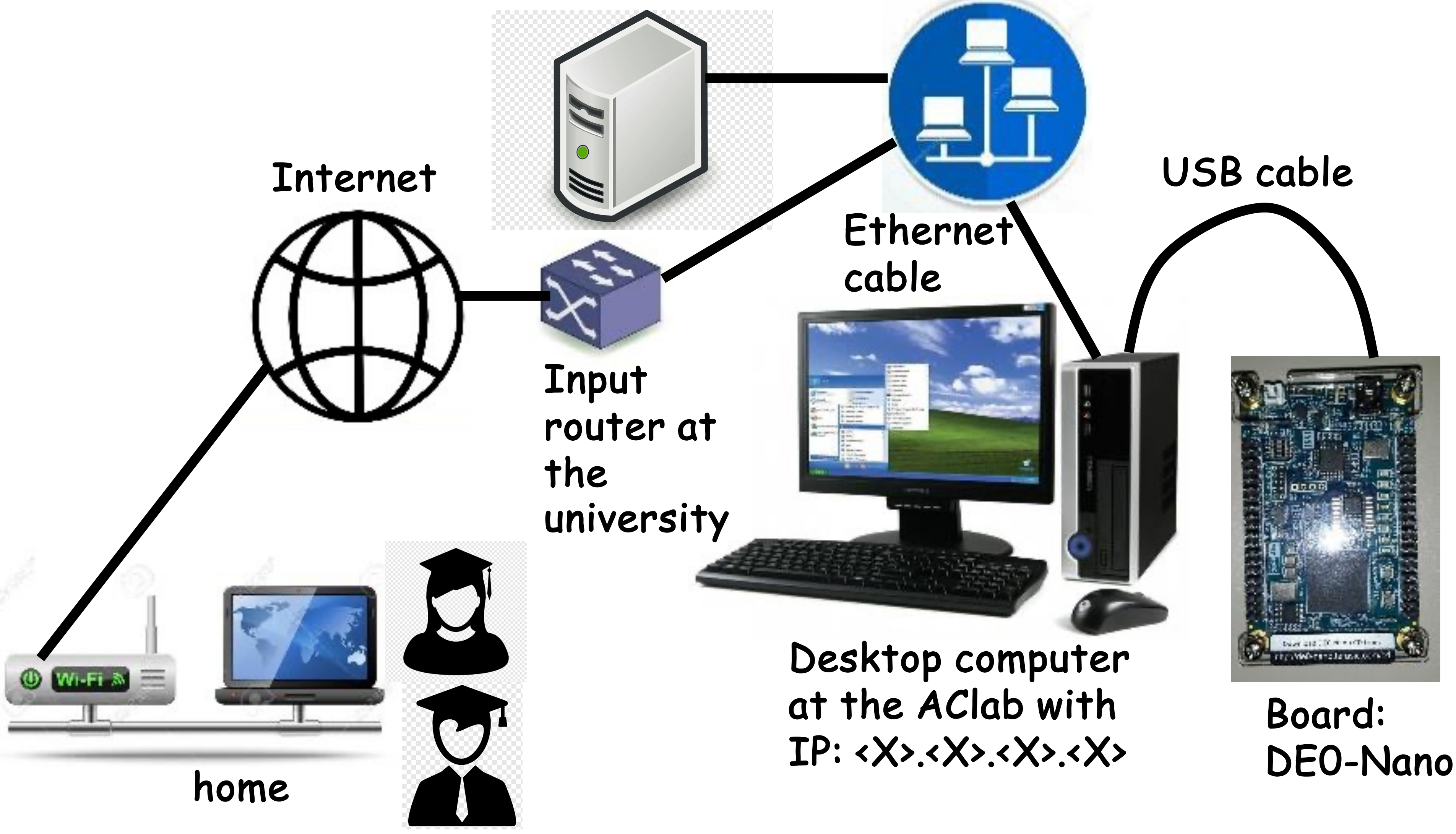
RISC-V instruction set architecture and programming of Nios V/m processor

Computer Architecture (40969)
Computer Science School (EII)
University of Las Palmas de Gran Canaria

Summary

- Hierarchy of computer abstraction levels
- Elements of RISC-V instruction set architecture
- Nios V soft processor operating modes
- General purpose and control registers
- Access to the address apace
- Types of RISC-V instructions
- Example of a RISC-V assembler program
- Subroutines

Remote Computer Architecture Laboratory



VPN server

Local LAN network at the Computer Science building

Internet

USB cable

Ethernet cable

Input router at the university

Desktop computer at the AClab with IP: <X>.<X>.<X>.<X>

Board: DE0-Nano

home

Example of a RISC-V assembler program

```
        .text                                /* executable code follows */
        .global  _start
_start:
        la       x16, 0x10000010             /* green LED base address */
        la       x15, 0x10000040             /* slider switch base address */
        la       x17, 0x10000050             /* pushbutton KEY base address */
        la       x19, LEDG_bits
        lw       x6, 0(x19)                  /* load pattern for LEDG lights */
DO_DISPLAY:
        lw       x4, 0(x15)                  /* load slider (DIP) switches */
        lw       x5, 0(x17)                  /* load pushbuttons */
        beq      x5, x0, NO_BUTTON
        mv       x6, x4                      /* use SW values on LEDG */
        add      a0, zero, x4
        add      a1, zero, 8
        jal      ra, rotl
        add      x4, a0, zero
        or       x6, x6, x4
        add      a0, zero, x4
        add      a1, zero, 8
        jal      ra, rotl
```

Example of a RISC-V assembler program

```
        add     x4, a0, zero
        or      x6, x6, x4
        add     a0, zero, x4
        add     a1, zero, 8
        jal     ra, rotl
        add     x4, a0, zero
        or      x6, x6, x4
WAIT:
        lw      x5, 0(x17)              /* load pushbuttons */
        bne     x5, x0, WAIT            /* wait for button release */
NO_BUTTON:
        sw      x6, 0(x16)              /* store to LEDG */
        add     a0, zero, x6
        add     a1, zero, 1
        jal     ra, rotl
        add     x6, a0, zero
        li      x7, 150000              /* delay counter */
DELAY:
        addi    x7, x7, -1
        bne     x7, x0, DELAY
        j       z0_DISPLAY
```

Example of a RISC-V assembler program

```
rotl:
        sll     a2,   a0, a1
        sub     a4, zero, a1
        srl     a3,   a0, a4
        or      a0,   a2, a3
        ret

        .data                               /* data follows */
LEDG_bits:
        .word   0x0F0F0F0F
        .end
```