



Lab Assignment 3:

Performance evaluation of pipelined processors

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

Main goals of Lab 3

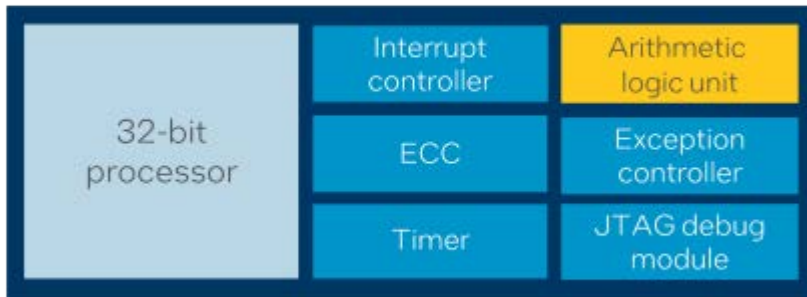
- Performance evaluation of the Nios V/g pipelined processor and comparison with the Nios V/m pipelined processor: CPI, operations/sec.
- Analyze the effect on Nios V/g performance of the machine instruction reordering software technique.
- Propose the realization of a theoretical exercise in which a possible change in the Nios V/g microarchitecture is evaluated.

Scheduling: 4 weeks

- S1: Section 1; 2 hours
- S2: Section 2; 2 hours
- S3: Sections 3 y 4; 2 hours
- S4: Test; 1.5 hours

Soft processors: Nios V/{m,g}

Nios V/**m**



Multicycle: minimum 4 clk/inst

Pipelined: 5 pipes

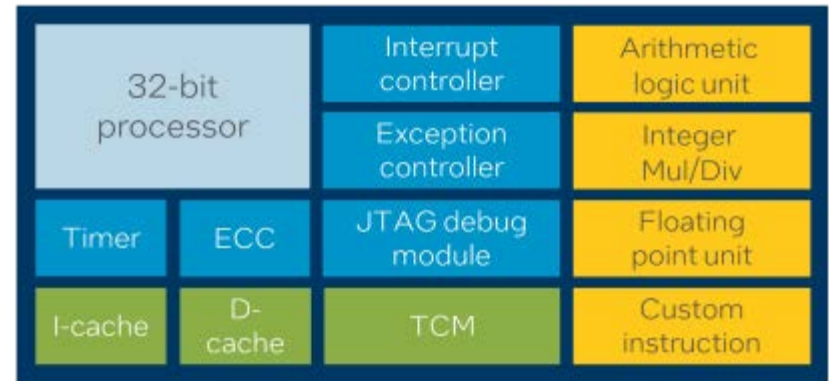
WITHOUT instruction cache

WITHOUT data cache

f = 50 MHz

Configuration file of the pipelined version for the DE0-Nano board:
"DE0_NanoBasic_Computer_22jul24.sof"
"

Nios V/**g**



Pipelined: 5 pipes

WITH instruction cache

WITH data cache

WITH dynamic branch prediction

f = 50 MHz

Configuration file for the DE0-Nano board:
"DE0_NanoBasic_Computer_23jul24.sof"

Instruction phases for the “multicycle” Nios V/m

Table 21. Processor Non-pipelined Stages

Stage	Denotation	Function
F	Instruction fetch	Pre-decode for register file read
D	Instruction decode	<ul style="list-style-type: none">• Decode the instruction• Register file read data available• Hazard resolution and data forwarding
E	Instruction execute	<ul style="list-style-type: none">• ALU operations• Memory address calculation• Branch resolution• CSR read/write
M	Memory	<ul style="list-style-type: none">• Memory and multicycle operations• Register file write• Next PC logic• Branch redirection

- Issue: 1 instruction / cycle
- Commitment: 1 instruction / cycle
- Base CPI = 4 cycles
- Stalls: multicycle operations, data dependences

Pipes for the Nios V/{m,g} cores

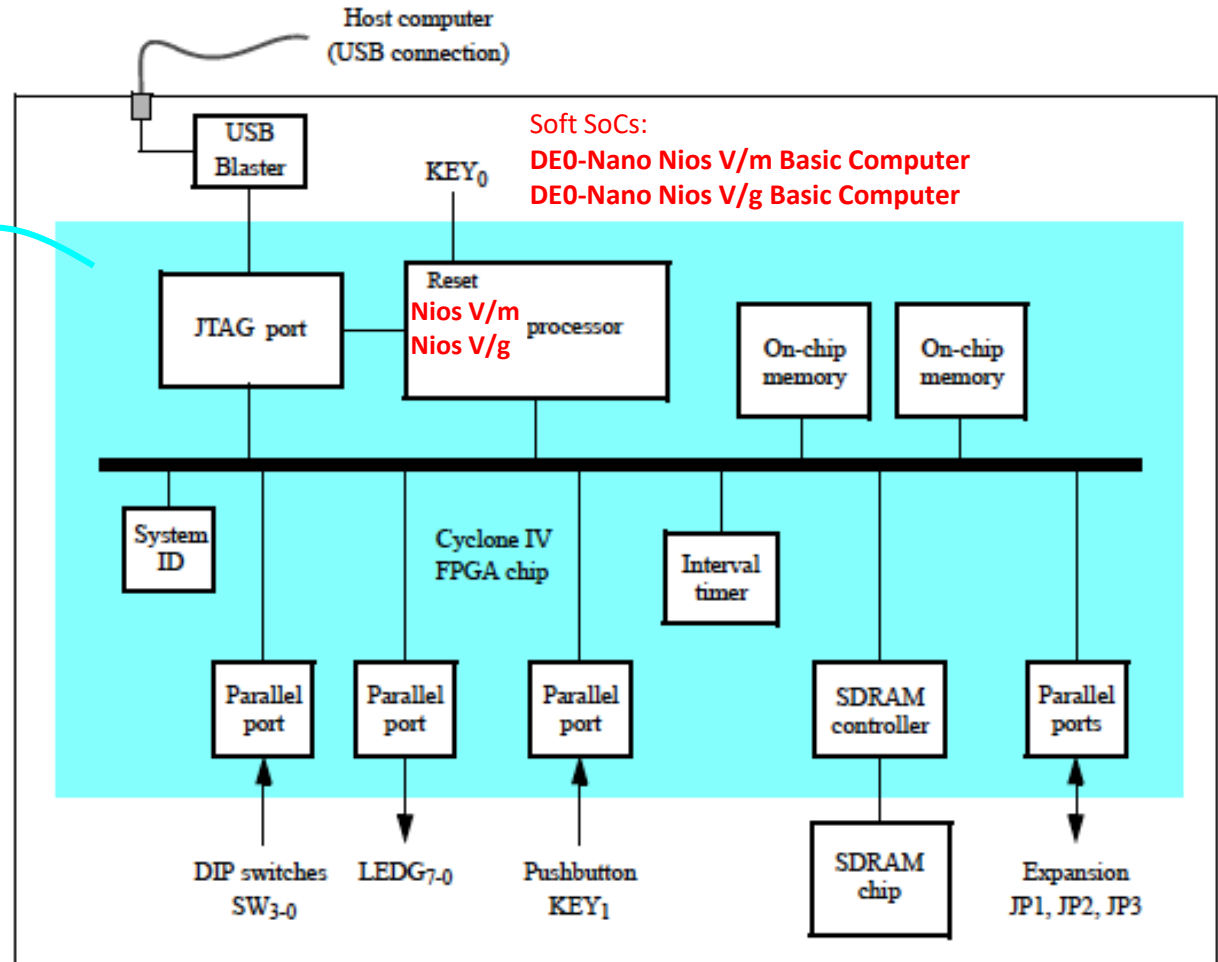
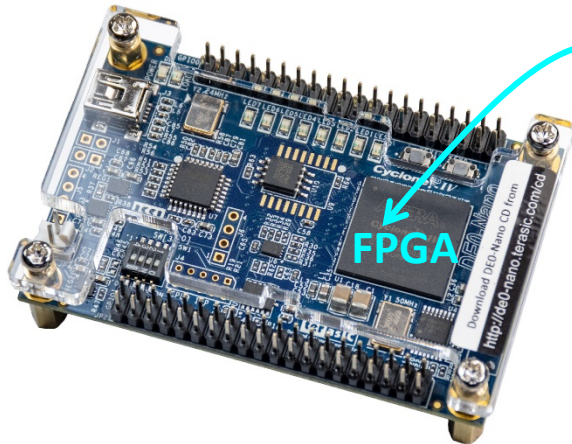
Table 63. Processor Pipeline Stages

Stage	Denotation	Function
F	Instruction fetch	<ul style="list-style-type: none">• PC+4 calculation• Next instruction fetch• Pre-decode for register file read
D	Instruction decode	<ul style="list-style-type: none">• Decode the instruction• Register file read data available• Hazard resolution and data forwarding
E	Instruction execute	<ul style="list-style-type: none">• ALU operations• Memory address calculation• Branch resolution• CSR read/write
M	Memory	<ul style="list-style-type: none">• Memory and multicycle operations• Register file write• Branch redirection
W	Write back	<ul style="list-style-type: none">• Facilitates data dependency resolution by providing general-purpose register value.

- Issue: 1 instruction / cycle
- Commitment: 1 instruction / cycle
- Base CPI = 1 cycle
- Stalls: multicycle operations, data dependences

Soft SoCs

Terasic DE0-Nano board

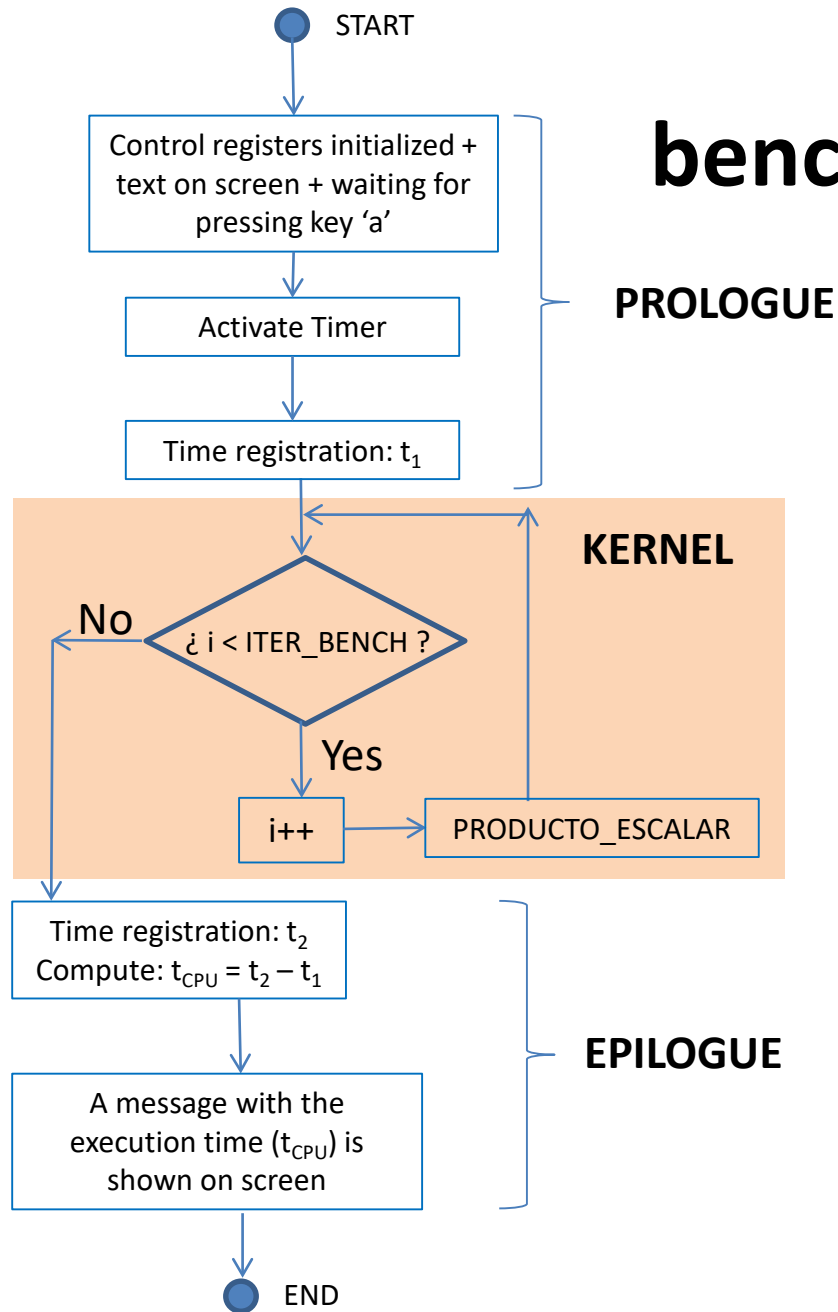


Part 1: Analysis of the usage of instructions types in a benchmark program and the CPI of Nios V/{m,g} processors

General Description. A synthetic benchmark program named *benchNIOSV2024_dotProduct* is used to **analyze the instruction usage of the 32-bit RISC-V instruction set**. This program performs the scalar product of two vectors repeatedly, as many times as specified by the program constant `ITER_BENCH`. Additionally, the CPI of the Nios V/{m,g} processors is calculated.

PART 1:

benchNIOSV2024_dotProduct



Parte 1: Objectives

- Objective 1: Classify the instructions by counting the number of times each instruction is executed in the PRODUCT_ESCALAR subroutine of the source code located in the file: `producto_escalar.s`. A flowchart of the main activities performed in the benchmark is shown in the transparency above.
- Objective 2: Next, calculate the total number of instructions executed and the percentage of each type of instruction (ALU, MEMORY, JUMP, OTHER) by filling in Table 1.
- Objective 3: Record the total number of clock cycles in which the benchmark is executed, both for the Nios V/m and Nios V/g processors, and calculate the CPI of the program. **Important:** Note that the non-kernel portion of the benchmark does not contribute a significant number of instructions to the CPI calculation.

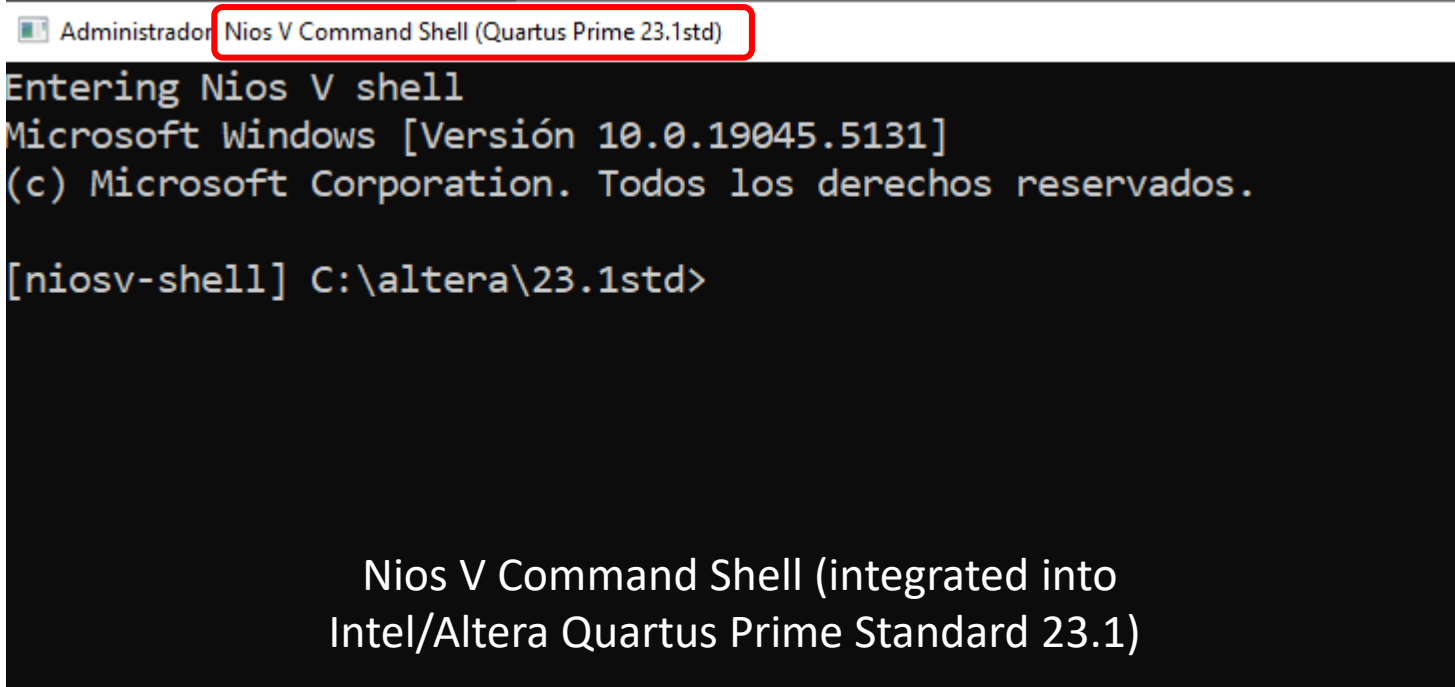
Part 1: hands-on exercises using Nios V/m

1. Create a new directory, for example: part1 and make cd. Execute the following command in the Nios V Command Shell Window.
2. Copy the files: `benchNIOSV2024_dotProduct.s`, `productoEscalar.s`, `escribir_jtag.s`, `DIV.s`, `BCD.s`, `Makefile` in this directory
3. Compile and link the benchmark program:
`benchNIOSII_dotProduct.`
4. Configure the FPGA in the DE0-Nano board using the file `DE0_NanoBasic_Computer_22jul24.sof` into the board DE0-Nano
5. Execute step by step to count the type of instruction.

Part 1: hands-on exercises using Nios V/g

The same steps as for Nios V/m except the configuration file for the DE0-Nano board:

DE0_NanoBasic_Computer_23jul24.sof



The screenshot shows a Windows command prompt window titled "Administrador Nios V Command Shell (Quartus Prime 23.1std)". The window has a black background with white text. The text inside the window reads: "Entering Nios V shell", "Microsoft Windows [Versión 10.0.19045.5131]", "(c) Microsoft Corporation. Todos los derechos reservados.", and "[niosv-shell] C:\altera\23.1std>".


```
Administrador Nios V Command Shell (Quartus Prime 23.1std)
Entering Nios V shell
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

[niosv-shell] C:\altera\23.1std>
```

Nios V Command Shell (integrated into
Intel/Altera Quartus Prime Standard 23.1)

Nios V Command Shell

- `$ cd <project directory>`
- `$ sh`
- `$ make help --> help for Makefile`
- `$ jtagconfig.exe`
- (modify Makefile: `SOFfile`, `LINKfile`)
- `$ make --> compila & link, output:`
`"benchNIOsv2024_dotProduct.elf"`
- `$ make configure`
- `$ make download`
- `$ juart-terminal.exe`



```
C:/altera/12.1sp1/University_Program/NiosII_Computer_Systems/DE0-Nano/DE0-Nano_Basic_Computer_Ni
osVm_conSDRAM/verilog/software/niosv/ACpractica3niosv/Part1 # juart-terminal.exe
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "USB-Blaster [USB-0]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

PRESS KEY A TO START THE BENCHMARK: A
... RUNNING ...
CYCLE COUNTER (CLK@50mHz): 0000000021225909
NUMBER OF ITERATIONS DONE: 00005000
bye!
```

Parte 1: Table 1

ALU instructions	Number of executions	MEMORY instructions	Number of executions	BRANCH & JUMP instructions	Number of executions	OTHER instructions	Number of executions
addi		lw		beq		nop	
...		
...		
Total ALU instructions		Total MEMORY instructions		Total BRANCH & JUMP instructions		Total OTHER instructions	
N (total number of executed instructions)							
% ALU		% MEMORY		% BRANCH & JUMPS		% OTHER	
Cicles Nios V/m				Cicles Nios V/g			
Total CPI of program Nios V/m				Total CPI of program Nios V/g			