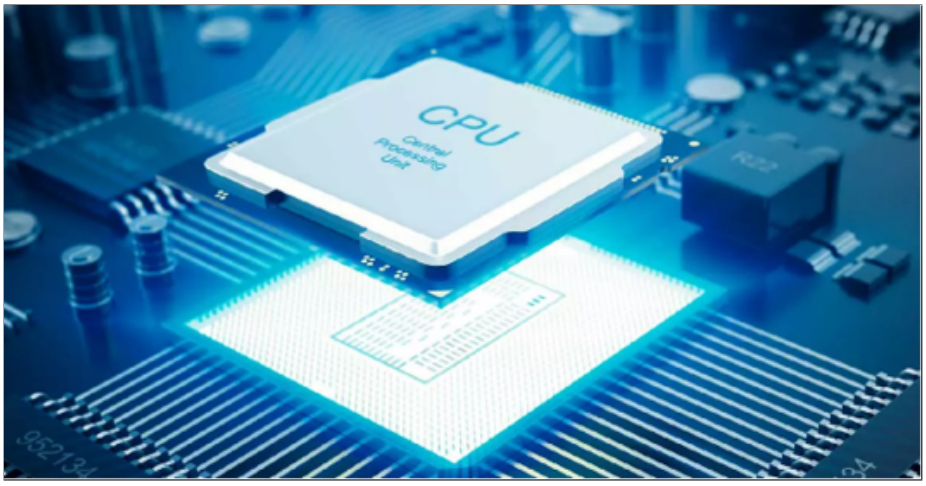
Computer Architecture - Lab Assignment 3*

Performance evaluation of pipelined processors



*AC 40969 - https://www.english.ulpgc.es/

Index

1. Introduction

The principal objective if this practice 3 consist in the evaluation of the segmented processor[1] performance Nios II/f and compare with the multicycle processor Nios II/e[2]. Additionally, we analyze the effect of the performance of Nios II/f of the software technique of reorder the instructions. Finally, we propose that realize a theory exercise where we evaluate the change of the microarchitecture Nios II/f using the data obtained during the practice activity.

This practice is divided into 4 parts which are the following

Part 1. You will do an analyze of the instructions execute in a benchmark program to know the percentage of each one of the types of the instructions: ALU, MEMORY, JUMP.

Part 2. You will do an analyze of the performance of the multicycle processors Nios II/e and segmented Nios II/f to know in which circumstances of the execution of the program is limited by the Access to memory or by the operations ALU. Additionally, we analyze the deteriorated of the performance of these processors that introduce the instructions of JUMP.

Part 3. We compare the effect of the performance of the segmented processor Nios II/f that produce the software technique of reorder the instructions.

Part 4. We propose you to evaluate the new design of the segmented processor.

The material in this practice you will find in the following files:

- benchNIOSII2021_Parte1
- benchNIOSII2021_Parte2
- benchNIOSII2021_Parte3
- N2fdCache512B-4bytes
- This document

You should use the Intel DE0-Nano board that it is found in one of the laboratories of the ULPGC (see Figure 1). To do this, each student should connect this board in a computer of the laboratory and execute in a virtual machine "Altera" where you find this tool Altera Monitor Program (AMP) that allows interacting with the board.

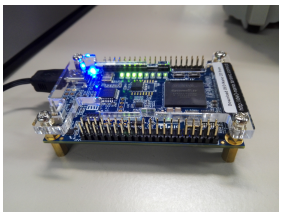


Figure 1: De0-Nano board

1.1. Analize the mix of types of instructions in a benchmark

General description: you use a benchmark synthetic benchNIOSII2021_Parte1 to analyze the mix of instructions of the instructions set Nios II of 32 bits. This program realizes the dot product of two vector many times, as indicated by the constant ITER_BENCH (see the subroutine PRODUCTO_ESCALAR in the file producto_escalar. s).

Objective 1: Classify the instructions, count the number of times that executes each one in the subroutine PRODUCTO_ESCALAR of the source code that its find in the file: producto_scalar. s. In this Figure 2. 1 it shows the flowchart of the principal activities that its realize in the benchmark.

Objective 2: Now calculate the total number of instructions execute and the percentage of each one type of instruction (ALU, MEMORY, JUMP and others) writing on the table 1.

Objective 3:Register the total number of clock cycles that execute the benchmark,for the both processors Nios II/e and Nios II/f, and calculate the CPI of the program of this processor. Important: Consider that the part of the benchmark that is not the compute kernel does not contribute a significant number of instructions to calculate the CPI.

Practice methodology with the processor multicycle Nios II/e:

1. Create a new Project in AMP using the configuration D0-Nano Basic Computer.
2. Write the shift of the sections .text and .data inside the address space in 0x400.
3. Compile and charge the benchmark program(file: benchNIOSII_Parte1) en in the board DE0-Nano.
4. Execute step by step to count the type of instructions that execute in the kernel of compute that you identify in the Figure 1 using breakpoints in the zone corresponding with the executable program.

Practice methodology with the processor multicycle Nios II/f:

1. Create a new Project in AMP using the configuration Custom System with the file sopcinfo: nios_system.sopcinfo and the file sof: DE0_Nano_Basic_Computer.sof(both are in the file: N2fdCache512B-4bytes).
2. Write the shift of the sections .text and .data inside the address space in 0x400.
3. Compile and charge the benchmark program(file: benchNIOSII_Parte1) en in the board DE0-Nano.
4. Execute the program until receiving the message “end of the program.

Files of this practice:

- Principal program of the benchmark: benchNIOSII2021_Parte1.s
- Subroutines: DIV.s, JTAG2021.s, productoEscalar.s, BCD.s, nios_macros.s (file: benchNIOSII_Parte1).
- Files of configuration of the NIOSII/f: nios_system.sopcinfo, DE0_Nano_Basic_Computer.sof (file: N2fdCache512B-4bytes)

Questions:

1. ¿In what type of program you classify the program bechNIOSII2021_Parte1 (arithmetic,dominated by memory Access o dominated by jump instructions)? Justify and argue the answer.
2. With the knowledge of the instructions that take on average in execute of the processor Nios II/e a total of 6 cycles[1], and that the processor Nios II/f[1]: 1 cycle(ALU), 1 cycle(MEMORY),2 cycles(JUMPS), obtained the theory CPI of the program in both processors.
3. What it is the differences that you find with the values obtained with the CPI about the timing in the Objective 3 of this part 1 in the DE0-Nano? ¿What cause you think that provoke these differences? Justify and argue the answer.

Figure 2. The flowchart of the benchmark program whose principal program, it is found in the file benchNIOSII2021_Parte1.s.

The benchmark it is divided in 3 parts: Preamble, compute of kernel and foreword.

In the preamble, we wait to introduce 'a' into the keyboard, configure the device of input/output of the Timer of the DE0-Nano board and register the first period of time t1.

In the compute kernel repeat many times the dot product of 2 vectors of 6 components in each vector.

In the foreword register the second period of time t2, calculate the time between the 2 periods of times and show the value into the terminal AMP.

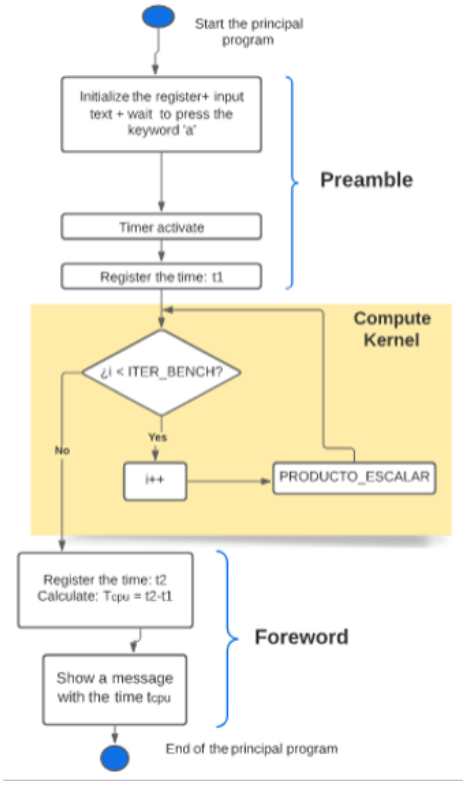


Table 1. Obtained the percentage of instructions execute by the processor Nios II/e,f using the subroutine PRODUCTO_ESCALAR that you find in the file producto_escalar.s(file: benchNIOSII_Parte1).

ALU Instruction	Number of executions	Memory Instructions	Number of executions	Jump Instructions	Number of executions	Others Instructions	Number of executions
addi		ldw		beq		nop	
...		...		...		...	
...		...		...		...	
Total Instructions ALU		Total Instructions MEMORY		Total Instructions JUMPS		Total Instructions OTHERS	
N (total instructions executed)							
% ALU		% MEMORY		% JUMPS		% OTHERS	
CYCLES NIOS II/e				CYCLES NIOS II/f			
CPI TOTAL OF THE PROGRAM NIOS II/e				CPI TOTAL OF THE PROGRAM NIOS II/f			

1.2. Analize the limitations of the relation "operations ALU/seg" in the execution of the program benchmarks in Nios II/e,f

General description: In this part you evaluate the limit of each software processor Nios IIe,f, about the number of arithmetic operations that you can do in each unit of time. Concrete will be analyzed the limit timings in "ALU/-seg operations" by the unit functional ALU, and the hierarchy of the memory and the jump instructions. To do this you use a second benchmark benchNIOSII2021_Parte2.

Objective1. Obtained the curve "operationsALU/seg" vs "operationsALU/byteMEMORY"(see figure 2). This curve represents the nivel of the performance of the processor Nios II timed in number of arithmetic operations by unit of time. In this curve, we can distinguish three parts(see Figure 2):

1. Zone 1 The peformance of the processor timed in "operationsALU/seg" are limited by the Access to memory(memory-bound). This zone it is characterised because incremented the number of arithmetic operations(operationsALU) by a byte of memory access, increment the number of arithmetic operations by second that the processor release.

2. Zone 2 The peformance of the processor timed in "operationsALU/seg" are limited by the number of arithmetic operations that reléase the processor per second(compute-bound). This zone is characterized because it is incremented the number of operationsALU by the byte of access memory, the number of

operations ALU per second is constant at the max level that it is possible for the processor.

3. Zone 3 the peformance of the processor in ALU operations/seg are deteriorated by the instructions Jump respect the max value of the zone 2. This zone is characterized because increment progressively the number of operations ALU per byte of memory accessed, the value of "operationsALU/seg" keeps constant, but in the level lower than the provided in the zone 2.

The methodology of the processors Nios IIe,f in this practical experience of the laboratory: Modify the source code that you find in the file roofline.s of the ways that we talk later.This code belongs to benchmark benchNIOSII2021_Parte2 (see Figure 3). Consecutively, the program is execute using both processor Nios II many times, as well as lines of the Table 2:

- Each line of the Table 2 has associated the execute of the benchmark program benchNIOSII2021_Parte2 in which previously should modify each one of the zones marked with comments in the file roofile.s
  - Modification of the Zone 1: active/uncomment as many ldw instructions as the column indicate in the table 2
  - Modification of the Zone 2: active/uncomment as many add instructions as the column "ALU" indicate in the table 2.
  - Modification of the Zone 3: active/uncomment as many iterations as the column "br" indicate in the table 2.
- Write in the table 2 for the processor Nios IIe,f.
- Represent the tuples of values(operationsALU/seg, operationsALU/byte-MEMORY) in two graphics X-Y as the Figure 2, one for each processor, using the values obtained in the table 2.

Practice methodology with the processor multicycle Nios II/e:

1. Create a new Project in AMP using the configuration D0-Nano Basic Computer.
2. Write the shift of the sections .text and .data inside the address space in 0x400.
3. Compile and charge the benchmark program(file: benchNIOSII_Parte2) en in the board DE0-Nano.
4. Execute step by step to count the type of instructions that execute in the kernel of compute that you identify in the Figure 1 using breakpoints in the zone corresponding with the executable program.

Practice methodology with the processor multicycle Nios II/f:

1. Create a new Project in AMP using the configuration Custom System with the file sopcinfo: nios_system.sopcinfo and the file sof: DE0_Nano_Basic_Computer.sof.
2. Write the shift of the sections .text and .data inside the address space in 0x400.
3. Compile and charge the benchmark program(file: benchNIOSII_Parte2) en in the board DE0-Nano.
4. Execute the program until receiving the message "end of the program".

Files of the practice:

- Benchmark program: benchNIOSII2021_Parte2.s, DIV.s, JTAG2021.s, roofline.s, BCD.s, nios_macros.s (file: benchNIOSII_Parte2).
- Files of configuration of Nios II/f: nios_system.sopcinfo, DE0_Nano_Basic_Computer.sof (file: N2fdCache512B-4bytes).

Questions:

4. From what number of operations per byte accessed in memory the processor Nios II/e, f are limited by the ALU operations? Which is the number max of ALU operations per seconds that can reach the processor Nios II/e? And the Nios II/f?
5. ¿Which is the max percentage of deterioration of the performance of the processors Nios II e, f when are executed the jump instructions?
6. ¿Are the benchmark program of the Part 1 (benchNIOSII2021_Parte1/producto_escalar) limited by memory or limited by the ALU operations?

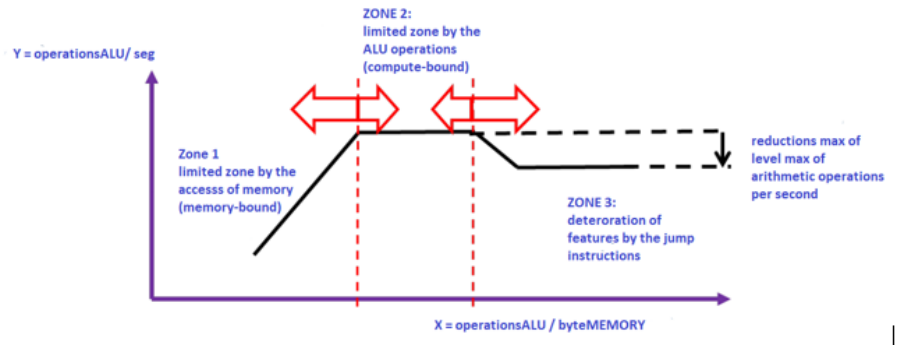


Figure 2: Curve "roofline" to determine the zones of the limitations of the ALU operations per unit of time.

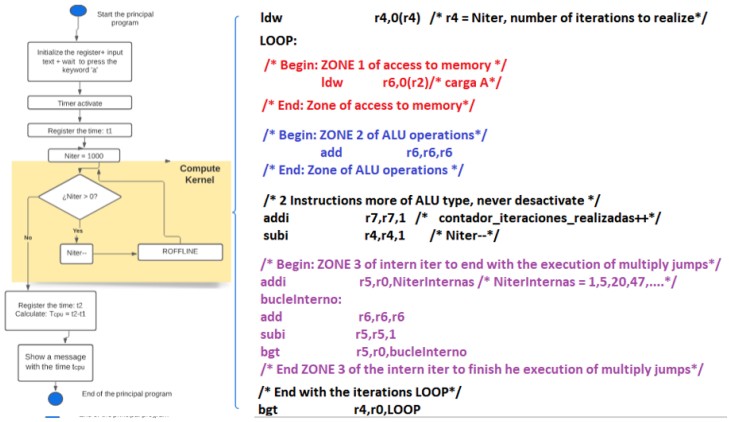


Figure 3: The flowchart of the benchmark program benchNIOSII2021_Parte2 and the intern iterations of the subroutine ROOFLINE that it is find in the file roofline.s.

Table 2. Obtain the curve "roofline" of limitation of the performance of the processors Nios II/e,f(see Figure 2). The cord x of the curve corresponds with the "operationsALU/byteMEM", and the cord Y corresponds with the "operationsALU/seg".

Important vocabulary:

- ID: number of identifications of the practice experiment to obtain the point of the curve.
- ldw: number of instructions ldw of the iterator LOOP of the source code roofline.s.
- ALU: number of instructions ALU of the iterator LOOP of the source code roofline.s
- Br: number of instructions Jump of the iterator LOOP of the source code roofline.s
- OperationsALU/byteMEM: number of instructions ALU that execute in each iteration of the iterator by each byte that it is accessed to the principal memory. The value is determined by the values of the correspond columns: ALU /4 * ldw.
- Niter: number of iterations of the iterator LOOP of the file roofline.s.
- OperationsALU/byteMEM: number of instructions ALU that execute in each iteration of the iterator by each byte that it is accessed to the principal memory. The value is determined by the values of the correspond columns: ALU /4 * ldw.
- Niter: number of iterations of the iterator LOOP of the file roofline.s.
- Tcpu: time execution of the program benchNIOSII2021_Parte2 and that is obtained with the equation: cycles/f, where f = 50 MHz 50 x 10 6 Hz. Tcpu it is timed in seconds.
- OperationsALU/seg: number of operations made in the functional unit ALU per unit of time. The value is obtained combining the values of the columns: Niter * ALU/ tcpu.
- CPI: Average number in which each instruction of the program its executes.The value is obtained combining the values of the columns: cycle/N.

Id of the curve point	**kernel**: roofline.s number of iterations by iterations			**Coord X** operations ALU/4byteMEM (ALU/4*ldw)	Iterations (N_{iter})	N (instructions excute	Nios II/f(e,f),f = 50 MHZ			
							cycles	t_{cpu} (seg=cycles/f)	**Coord y** operations ALU/seg (N_{iter} * ALU / t_{cpu})	CPI (cycles/N)
	ldw	ALU	br							
1	4	3	1		1000					
2	3	3	1		1000					
3	2	3	1		1000					
4	1	3	1		1000					
5	1	4	1		1000					
6	1	5	1		1000					
7	1	7	1		1000					
8	1	11	1		1000					
9	1	15	1		1000					
10	1	19	1		1000					
11	1	23	1		1000					
12	1	27	1		1000					
13	1	31	1		1000					
14	1	35	1		1000					
15	1	39	1		1000					
16	1	47	1		1000					
17	1	50	1		1000					
18	1	58	1		1000					
19	1	88	1		1000					
20	1	142	1		1000					
21	1	848	1		1000					
22	1	1048	1		1000					

1.3. Analize the effects of the reorder instructions in Nios II/f

General description: you use a new benchmark program synthetic benchNIOSII2021_Parte3 to evaluate the effect of the dependencies of data RAW between instructions of load and instructions ALU. This program benchmark are similar to the part 1 and 2 of this practice except, that is modify the kernel of computing that now it is found in a file bypassing.s. Later, it is proposed to apply the technique of reorder the instruction to reduce the time execution of the benchmark.

Objective 1: Edit the file bypassing.s and put the instructio add that is dependent on data with the instruction ldw(see version 1 in the figure 4). Later,timed and write the time of execution of the program benchmark that is obtained in the terminal of AMP using the processor Nios I/f.

Objective 2: Edit the file, bypassing.s and put the instruction add that is not dependent on data with the instruction ldw(see version 1 in the figure 4). Later, timed and write the time of execution of the program benchmark that is obtained in the terminal of AMP using the processor Nios I/f.

Questions:

7. Quantify and indicate the increment of time of execution of the program (tcpu) in percentage(%) when exist dependencies of data ldw -¿add respect to the non-existence of dependencies of data ldw -¿add.

8. Calculate the CPI of the two versions of the program benchmark bench-NIOSII2021_Parte3: version 1 and 2. Moreover count only the instructions execute in the iterator LOOP, observe that exist a constant Niter = 1000 in the source code of bypassing.s.

9. From the two values of CPI of the question 8, caculate the value of CPI and penalization due to the true dependecy RAW) and the percentage respect to CPI and the versión (without penalizations RAW). Remind that: CPI = CPIbase + CPI penalization.

10. Propose a new version(version 3) of the version 1 of the benchmark using the technique of reorder of instructions.Observe that does not have to change the number of instructions that its execute inside of iterator LOOP.

11. Create the version 3 of the benchmark program and timed the time of execution in AMP using the processor Nios II/f.

12. Calculate the speed-up of the new versión 3 respect to the versión 1.

13. Calculate the speed-up of the new versión 3 respect to the versión 2.

14. With the last results. What relative conclusión of the performance of the segmented processor Nios II/f you can obtain the application of the technique of reorder of instructions?

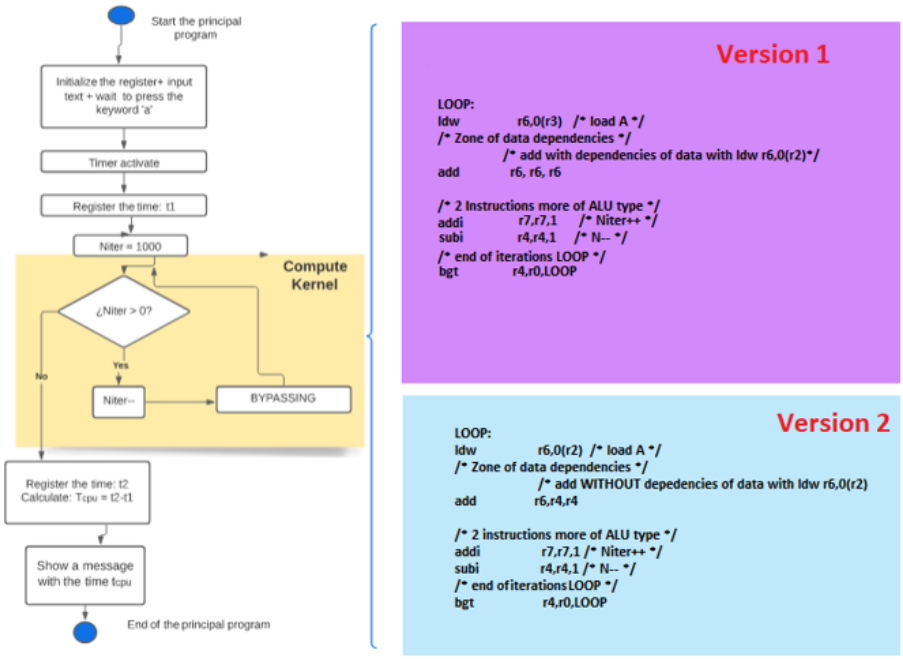


Figure 4: Flowchart of the benchmark program benchNIOSII2021_Parte3 and intern iterator LOOP of the subroutine BYPASSING that it is found in the file bypassing.s.

1.4. Problem of the design and new segmented processor

Imagine,that your boss of the departament of architecture of computers with Intel ask you to evaluate a possible modification of the design of the segmented processor Nios II/f of 6 stages. The modification proposes is that the stage "Execution/Calculate of address (E)" and the stage of "Access a memory(M)" merge in an only step[2]. In this stage combined, the ALU, and the memory it will work parallel. The instructions for access to the data using the memory,leaving the ALU inactive, and the instructions, arithmetic-logic using the ALU leaving the memory inactive. This change will be profitable in terms of are and eficient energy.

In the architecture of instructions set Nios II, the address efective of an instruction of load and stored is calculate adding the content of a register(rs1) with a inmediate value(imm). The problema with the new design is that now no calculation is done with direction in the instructions of load or store,since these instructions cant acces to the ALU.

The defensors of the new design of the processor Nios II try to change the architecture of the instructions set to allow only one mode of address, call addressing direct of the register. In this mode of addressing, only use a source register, and the value that have, is the address of memory which is accessed. You can't specify either shifting with immediates.

In Nios II, the only form to realize the addressing direct to the register is specifies the address in a register and assign the value 0 to immediate, inm = 0. With the propose design, any instruction on load or store that use the addressing register.Inmediate with imm ¡= 0, need 2 instructions. In the first place, the values of the register and the immediate should add an instruction addi, and then this address calculates can store or load in the following instruction. The instructions of load and store that actually using an inm= 0. Does not require instructions additionally in the new design.

Questions:

15. Your work consists in deciding the percentage of increase, the number total of instructions that would have to execute under the new design of 5 stages. This will require an analyzed more detailed of the different types of load and stores executes by the subroutine PRODUCTO_ESCALAR of the benmark benchNIOSII2021_Parte1 and was analyzed in the part 1 of this practice 3.

16. Evaluate the new design depending on the percentage increasing of the number of instructions that will have to execute.

17. What design recommend to your Intel boss that should do? Justify an answer quantitatively.

Complementary Bibliography

[1] Altera/Intel. Nios ii core implementation details. `https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/nios2/n2cpu_nii51015.pdf`, 2015.

[2] Altera/Intel. Nios ii processor reference guide. `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2cpu-nii5v1gen2.pdf`, 2020.