Lab Assignment 2:

Performance evaluation of the memory hierarchy of a computer and reverse engineering of the data cache memory

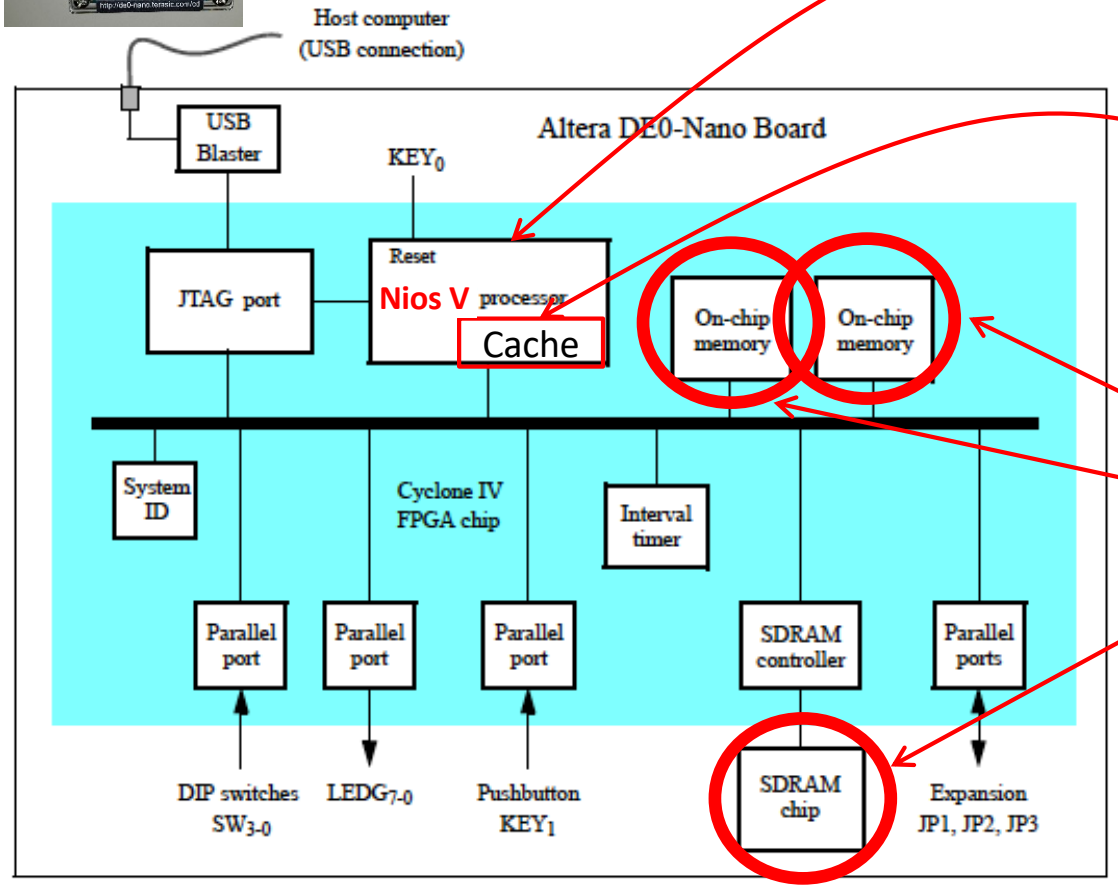
Computer Architecture (40969)
School of Computer Science (EII)
University of Las Palmas de Gran Canaria

Scheduling: 4 weeks

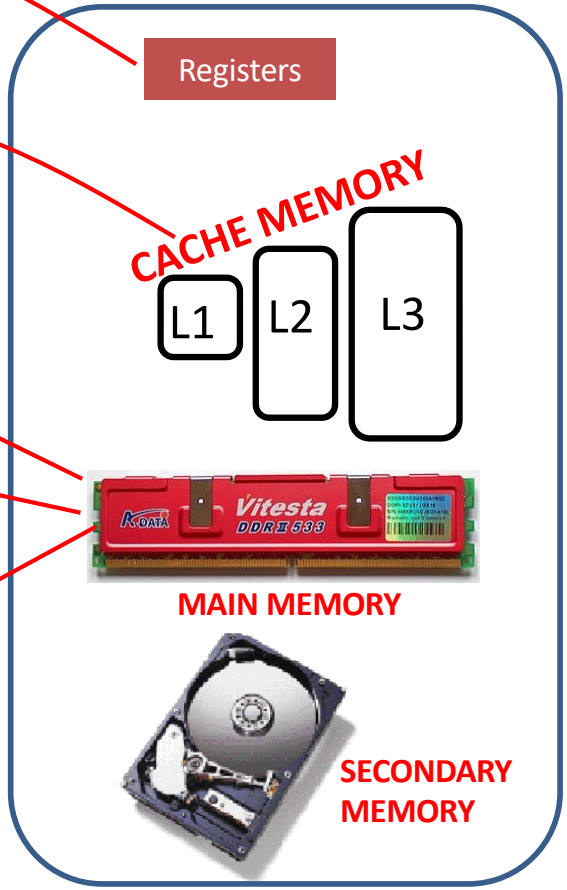
- Session 1: Activities 1,2,3; 2 hours
- S2: Activity 4 ; 2 hours
- S3: Reverse engineering for the data cache of Nios V/g; 2 hours
- S4: Examn; 1,5 hours

Implementing the Memory Hierarchy Levels of the Basic Computer Structure of the DE0-Nano board

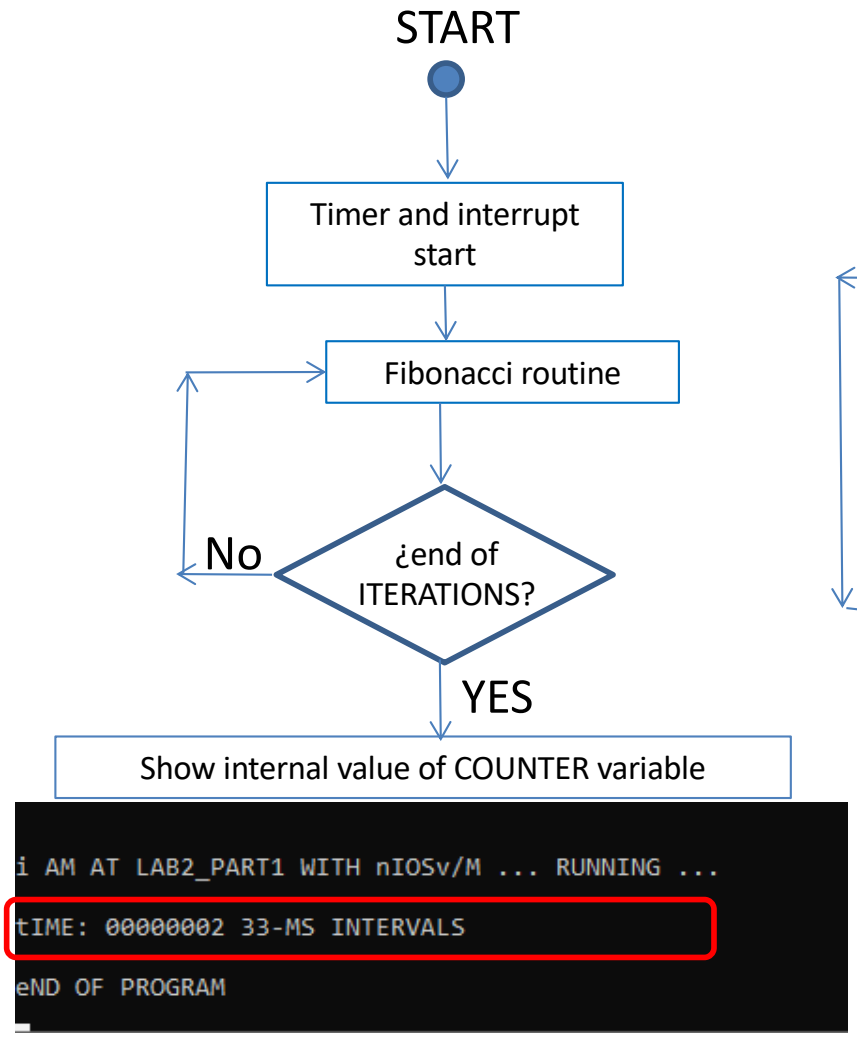
DE0-Nano

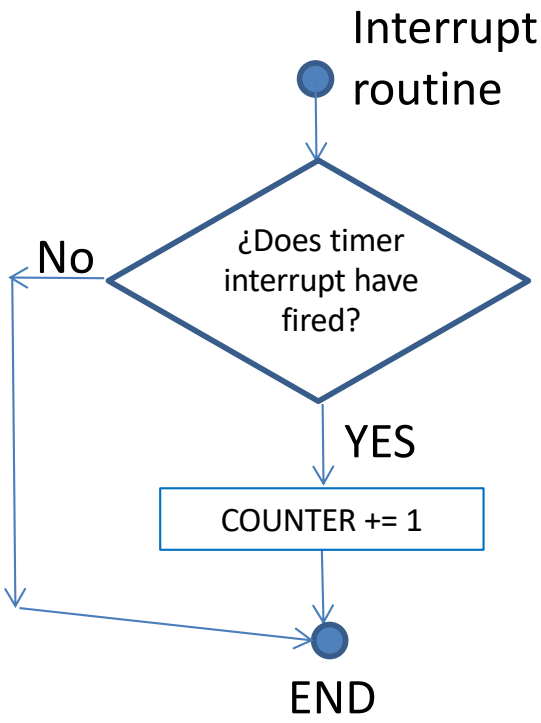


Implementation of the memory hierarchy



Activities 1,2,3: benchmark





Compiling & Linking using Nios V/m + SDRAM

```
C:/altera/12.1sp1/University_Program/NiosII_Computer_Systems/DE0-Nano/DE0-Nano_Basic_Computer_NiosVm_conSDRAM/verilog/so
ftware/niosv/ACpractica2niosv # make
riscv32-unknown-elf-as.exe  lab2_part1_2_3_main.s -alsg -o lab2_part1_2_3_main.s.obj > lab2_part1_2_3_main.s.log
riscv32-unknown-elf-as.exe  excepcionTimer.s -alsg -o excepcionTimer.s.obj > excepcionTimer.s.log
riscv32-unknown-elf-as.exe  escribir_jtag.s -alsg -o escribir_jtag.s.obj > escribir_jtag.s.log
riscv32-unknown-elf-as.exe  contador.s -alsg -o contador.s.obj > contador.s.log
riscv32-unknown-elf-as.exe  DIV.s -alsg -o DIV.s.obj > DIV.s.log
riscv32-unknown-elf-as.exe  BCD.s -alsg -o BCD.s.obj > BCD.s.log
riscv32-unknown-elf-as.exe  lab2_part1_2_3_fibo.s -alsg -o lab2_part1_2_3_fibo.s.obj > lab2_part1_2_3_fibo.s.log
riscv32-unknown-elf-ld.exe  -g -T linker_SDRAM.x  -nostdlib -e _start -u _start --defsym __alt_stack_pointer=0x08001F00
--defsym __alt_stack_base=0x08002000 --defsym __alt_heap_limit=0x8002000 --defsym __alt_heap_start=0x8002000  -o lab2_pa
rt1_2_3_main.elf lab2_part1_2_3_main.s.obj excepcionTimer.s.obj escribir_jtag.s.obj contador.s.obj DIV.s.obj BCD.s.obj l
ab2_part1_2_3_fibo.s.obj
riscv32-unknown-elf-ld.exe: warning: lab2_part1_2_3_main.elf has a LOAD segment with RWX permissions
niosv-stack-report  -p riscv32-unknown-elf- lab2_part1_2_3_main.elf
lab2_part1_2_3_main.elf
 * 1320 B - Program size (code + initialized data).
 * 256 B - Free for stack.
 * 0 B - Free for heap.
riscv32-unknown-elf-objdump  -Sdtx lab2_part1_2_3_main.elf > lab2_part1_2_3_main.elf.objdump
riscv32-unknown-elf-objcopy -O binary lab2_part1_2_3_main.elf lab2_part1_2_3_main.hex
C:/altera/12.1sp1/University_Program/NiosII_Computer_Systems/DE0-Nano/DE0-Nano_Basic_Computer_NiosVm_conSDRAM/verilog/so
ftware/niosv/ACpractica2niosv #
```

Executing using Nios V/m + SDRAM

```
[OpenOCD output] Info : Virtual Tap/SLD node 0x08986E00 found at tap position 0 vtap position 1
[OpenOCD output] Info : datacount=2 progbufsize=8
[OpenOCD output] Info : Examined RISC-V core; found 1 harts
[OpenOCD output] Info :  hart 0: XLEN=32, misa=0x4000b0c3
[OpenOCD output] Info : starting gdb server for tap_020F30DD.0.niosv_0.cpu on 0
[OpenOCD output] Info : Listening on port 49626 for gdb connections
INFO: Found gdb port 49626
[OpenOCD output] Ready for Remote Connections
[OpenOCD output] Info : tcl server disabled
[OpenOCD output] Info : Listening on port 49627 for telnet connections
INFO: Found telnet port 49627
INFO: OpenOCD is ready.
INFO: Sending reset halt to OpenOCD.
INFO: Loading image via GDB. Running "riscv32-unknown-elf-gdb -batch -ex set arch riscv:rv32 -ex set remotetimeout 60 -e
x target extended-remote localhost:49626 -ex load lab2_part1_2_3_main.elf -ex set $mstatus &= ~(0x00000088)".
The target architecture is set to "riscv:rv32".
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
0x09000004 in ?? ()
Loading section .exceptions, size 0x5c lma 0x0
Loading section .text, size 0x38c lma 0x5c
Loading section .rwdata, size 0x140 lma 0x3e8
Start address 0x0000005c, load size 1320
Transfer rate: 4 KB/sec, 440 bytes/write.
[Inferior 1 (Remote target) detached]
INFO: Sending resume to OpenOCD.
```

```
C:/altera/12.1sp1/University_Program/NiosII_Computer_Systems/DE0-Nano/DE0-Nano_Basic_Computer_NiosVm_conSDRAM/verilog/so
ftware/niosv/ACpractica2niosv # 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)


i AM AT LAB2_PART1 WITH nIOSv/M ... RUNNING ...

tIME: 00000008 33-MS INTERVALS

eND OF PROGRAM
```

Executing using Nios V/m + SRAM

```
[OpenOCD output] Info : Listening on port 53290 for telnet connections
INFO: Found telnet port 53290
INFO: OpenOCD is ready.
INFO: Sending reset halt to OpenOCD.
INFO: Loading image via GDB. Running "riscv32-unknown-elf-gdb -batch -ex set arch riscv:rv32 -ex set remotetimeout 60 -e
x target extended-remote localhost:53289 -ex load lab2_part1_2_3_main.elf -ex set $mstatus &= ~(0x00000088)".
The target architecture is set to "riscv:rv32".
warning: No executable has been specified and target does not support
determining executable automatically.  Try using the "file" command.
0x09000004 in ?? ()
Loading section .exceptions, size 0x5c lma 0x8000000
Loading section .text, size 0x3a8 lma 0x800005c
Loading section .rwdata, size 0x140 lma 0x8000404
Start address 0x0800005c, load size 1348
Transfer rate: 4 KB/sec, 449 bytes/write.
[Inferior 1 (Remote target) detached]
INFO: Sending resume to OpenOCD.
C:/altera/12.1sp1/University_Program/NiosII_Computer_Systems/DE0-Nano/DE0-Nano_Basic_Computer_NiosVm_conSDRAM/verilog/so
ftware/niosv/ACpractica2niosv # 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)


i AM AT LAB2_PART1 WITH nIOSv/M ... RUNNING ...

tIME: 00000002 33-MS INTERVALS

eND OF PROGRAM
```

Activities 1,2,3: measuring execution time

Soft processor model + memory technology	Execution time	Speed-up
Nios V/m + SDRAM memory (Activity 1)		1X
Nios V/m + on-chip memory (Activity 2)		
Nios V/g + SDRAM memory (Activity 3)		
Nios V/g + on-chip memory (Activity 3)		

Activity 4: discovering data cache microarchitecture: data size and block size

Source code for the main benchmark program:

`lab2_part1_2_3_main.s`

```
                ...
                la a4, ITERACIONES      /* initializes the LOOP iteration counter, each iteration executing a Fibonacci loop */
                addi  s2, zero, 0       /* initializes the interval counter of program "s2". */

LOOP:           beq   a4, zero, END     /* execute Fibonacci loop */
                jal  ra, FIBONACCI
                addi  a4, a4, -1
                j    LOOP
```

Source code for the NEW subroutine:

`lab2_part1_2_3_fibo.s`

```
add             s4, zero, zero
la              s5, X
la              s6, V
LOOP:
  bge           s4, s5, STOP
  add           s7, s6, s4
  lb            zero, 0(s7)
  addi          s4, s4, P
  j             LOOP
...
.data
V:              .skip 65536
...
```

Activity 4

Modify source code in the file::
`lab2_part1_2_3_fibo.s`

```
add     s4, zero, zero
la      s5, X
la      s6, V
LOOP:
  bge   s4, s5, STOP
  add   s7, s6, s4
  lb    zero, 0(s7)
  addi  s4, s4, P
  j     LOOP
...
.data
V:      .skip 65536
...
```

X: data size accessed by program,
X = P × E

E: number of accesses to main memory

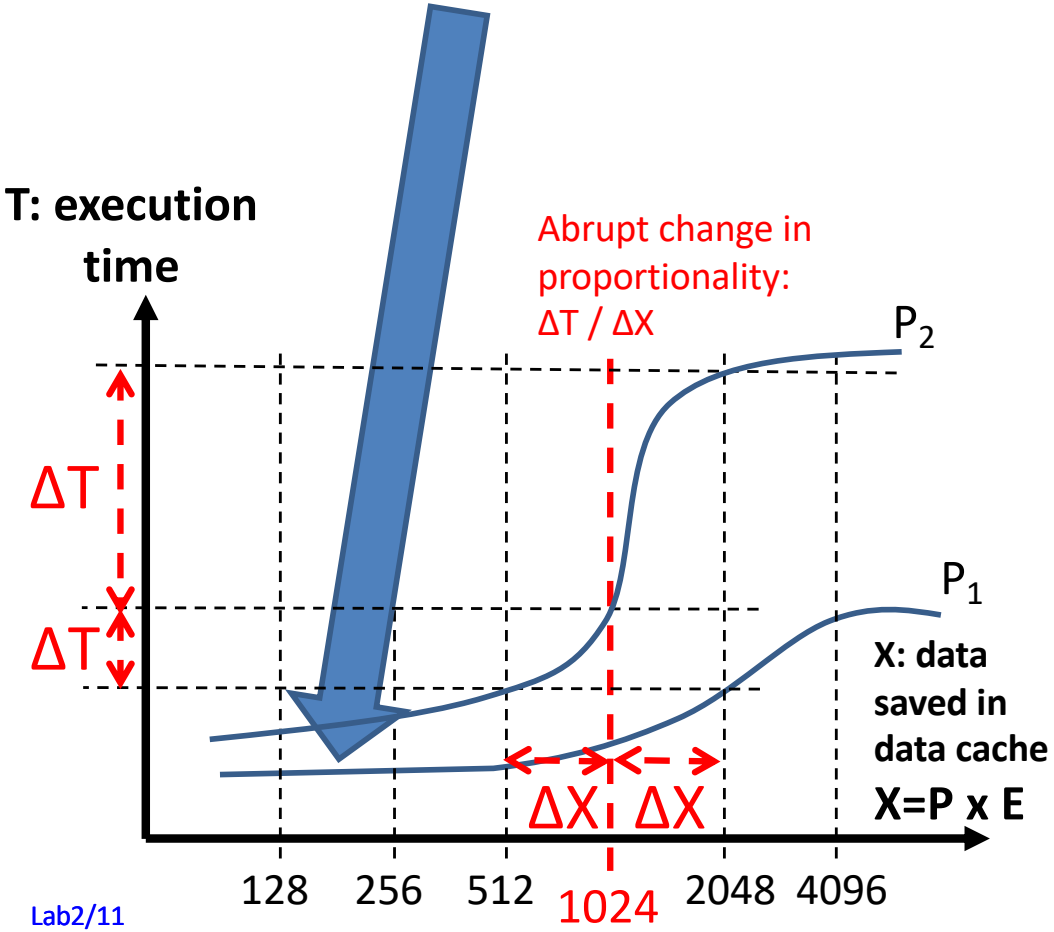
Table. Measures collected after running the benchmark for different data size and data access patterns

	E: Number of V vector bytes accessed actually					
P: data stride	128	256	512	1024	2048	4096
P = 1: every one						
P = 2: every two						
P = 4: every four						
P = 8: every eight						

Activity 4

Table. Measures collected after running the benchmark for different data size and data access patterns

	E: Number of V vector bytes accessed actually					
P: data access pattern	128	256	512	1024	2048	4096
P = 1: every one						
P = 2: every two						
P = 4: every four						
P = 8: every eight						



Method to fill in the table on the left

1) jtagconfig.exe
2) make configure

CHANGE PARAMETERS X AND P:

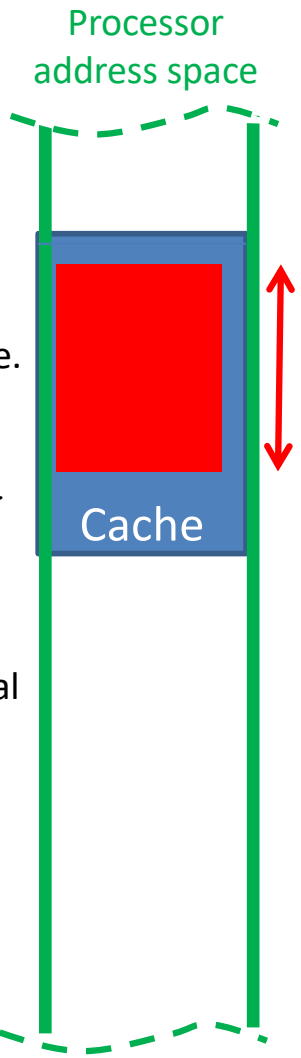
3) Modify the values for **X** and **P** in: `lab2_part1_2_3_fibo.s`
4) make
5) make download
6) juart_terminal.exe
7) register execution time

GO CHANGE PARAMETERS X AND P

¿Data cache size?: P=1

```
add     s4, zero, zero
la      s5, X
la      s6, V
LOOP:
  bge   s4, s5, STOP
  add   s7, s6, s4
  lb    zero, 0(s7)
  addi  s4, s4, P
  j     LOOP
...
.data
V:      .skip 65536
...
```

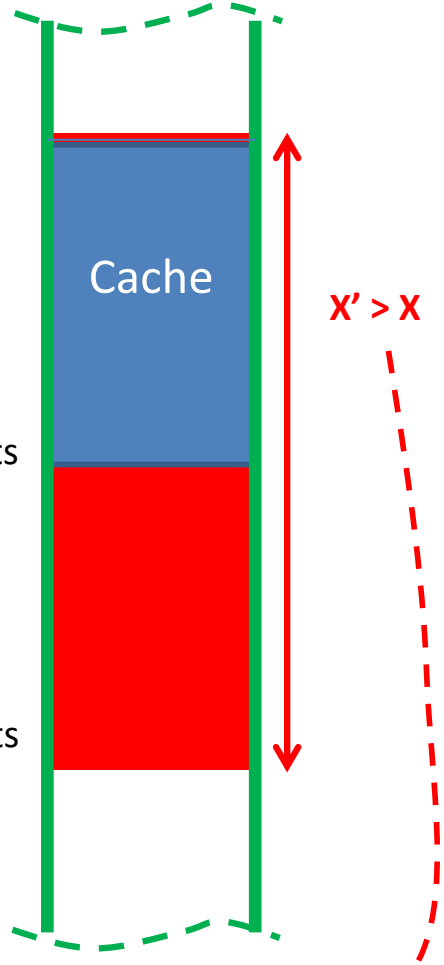
- X **does** fit in the data cache.
- There are only misses in ITERATION=1.
- In ITERATION={2,...,50000} the memory accesses hit the cache.
- Execution time (T) is approximately proportional to the number of memory accesses (E).



Case Type-1:
NO-overflow
X: data volumes handled by the program; the bytes of the V vector accessed from the program fit in the data cache.

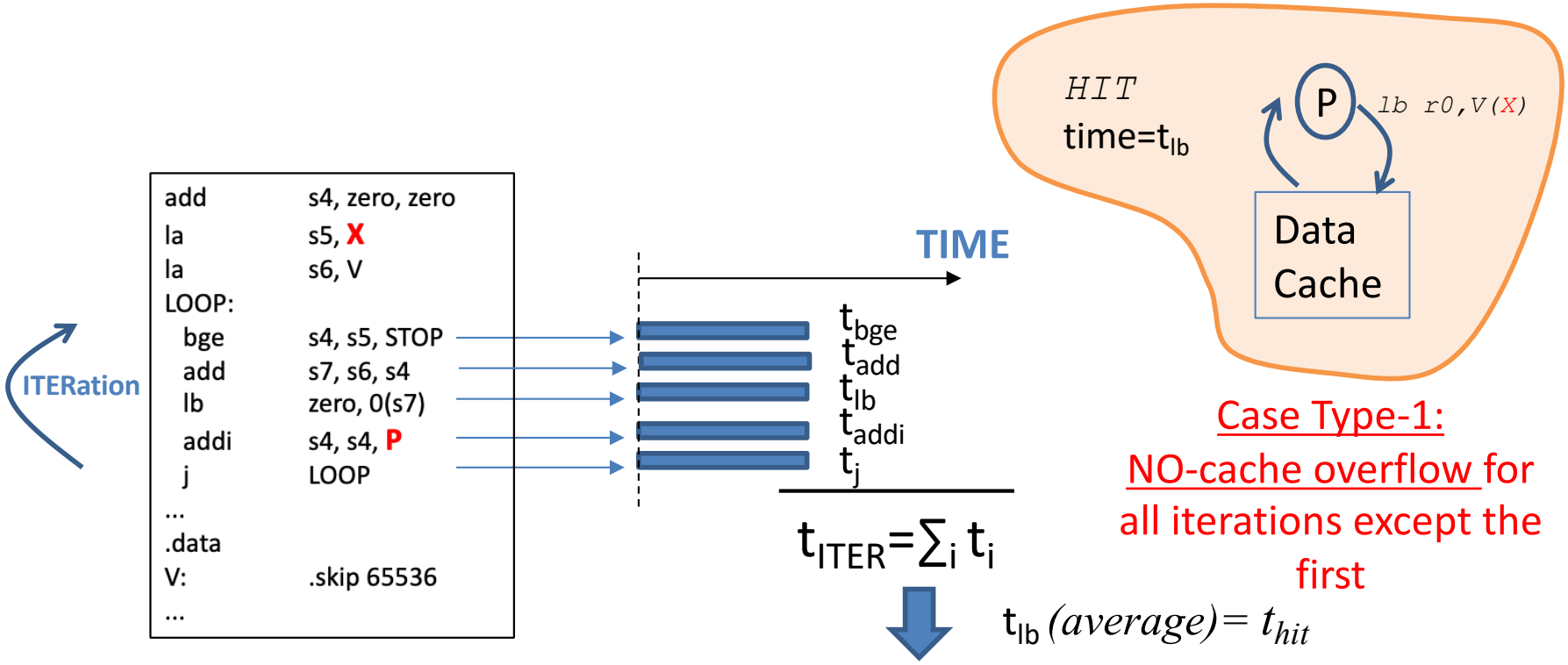
Case Tyte-2:
OVERFLOW

- **X' does not** fit in the data cache.
- Many failures in data cache by replacements in all ITERATIONS.
- Execution time does NOT have the same dependence on the number of memory accesses as when X fits in the data cache.



- **KEY:** find X' for P=1 which means that the execution time does not keep proportion with the number of accesses E as in lower X; the capacity is X'/2

Execution time **WITHOUT** cache misses after the first LOOP ITERATION



Case Type-1:
NO-cache overflow for all iterations except the first

$$X \rightarrow 2X$$

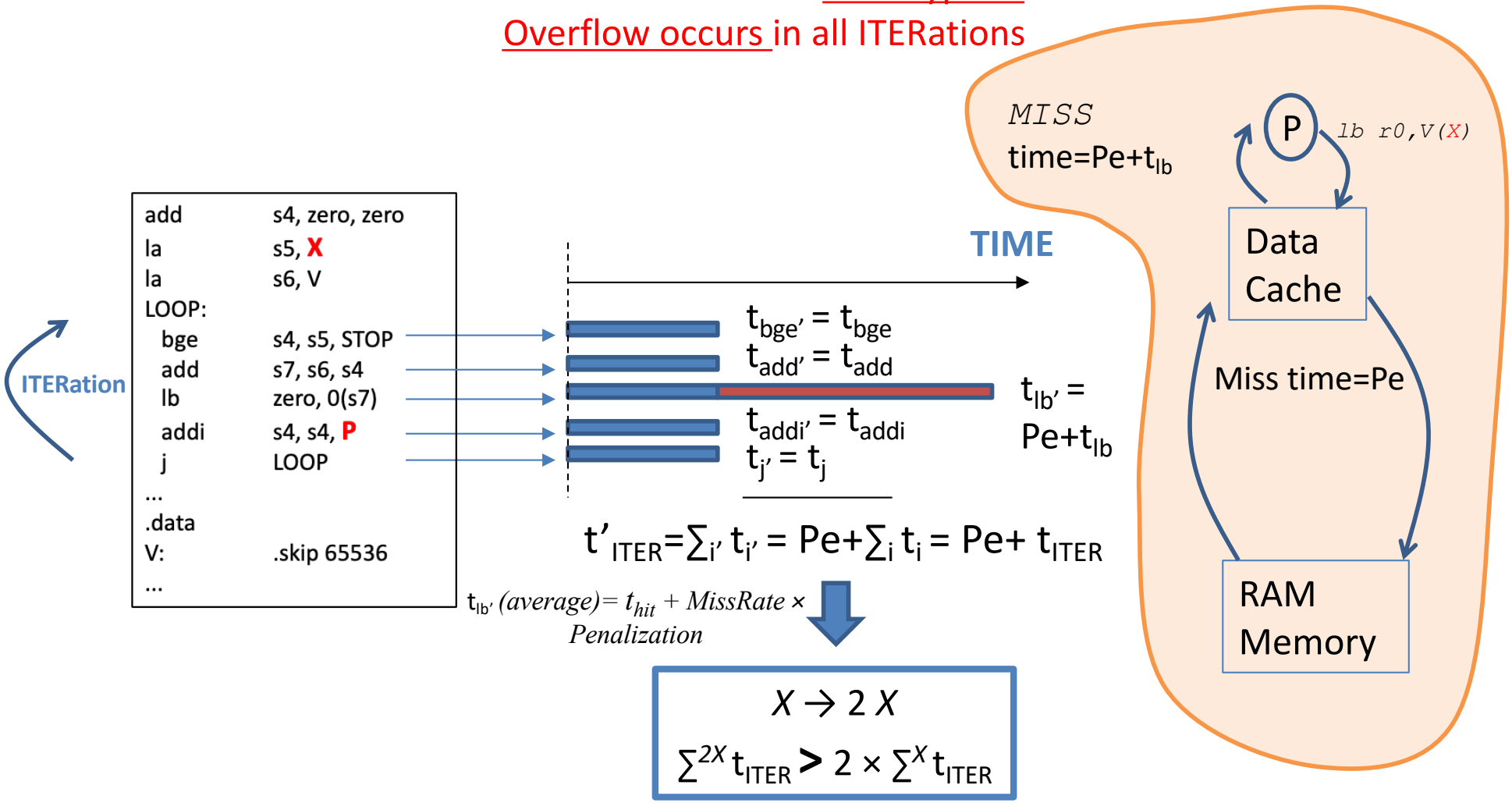
$$\sum^{2X} t_{ITER} \approx 2 \times \sum^{X} t_{ITER}$$

Double the number of ITERations (2X) should cause the program to take twice as long

Execution time **WITH** cache misses

Case Type-2:

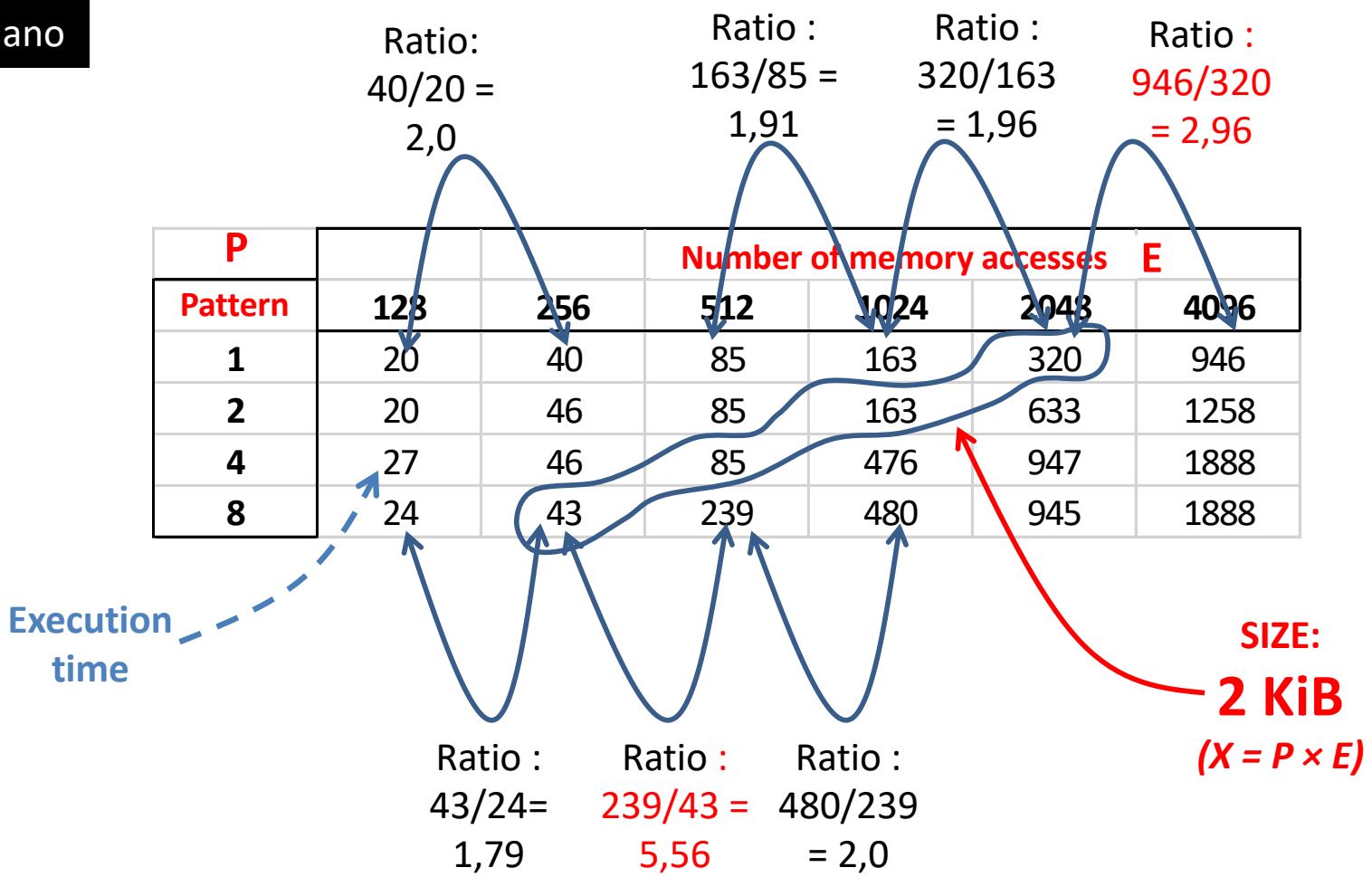
Overflow occurs in all ITERations



Double the number of ITERations (2X) should cause the program to take longer than **twice** the time with half the number of ITERations (X).

¿How do you discover cache size?: P=1

Board:
DE0-Nano



P	Number of memory accesses E					
Pattern	128	256	512	1024	2048	4096
1	20	40	85	163	320	946
2	20	46	85	163	633	1258
4	27	46	85	476	947	1888
8	24	43	239	480	945	1888

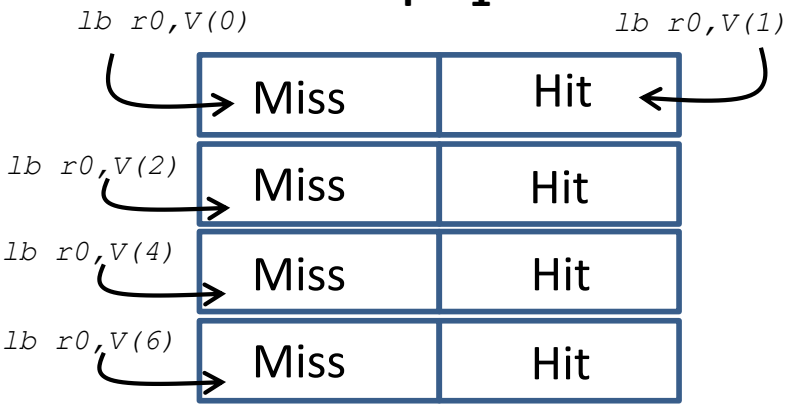
¿ How do you discover block size?

```
add     s4, zero, zero
la      s5, X
la      s6, V
LOOP:
  bge   s4, s5, STOP
  add   s7, s6, s4
  lb    zero, 0(s7)
  addi  s4, s4, P
  j     LOOP
...
.data
V:      .skip 65536
...
```

- Assume the case where the block size is 2 bytes.
- Assume that X is large enough so that the cache memory is always overflowing.

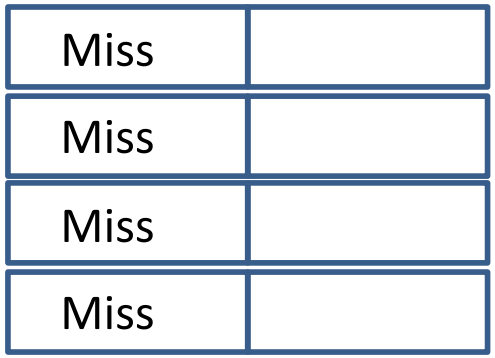
BLOCK: **2 bytes**

P = 1



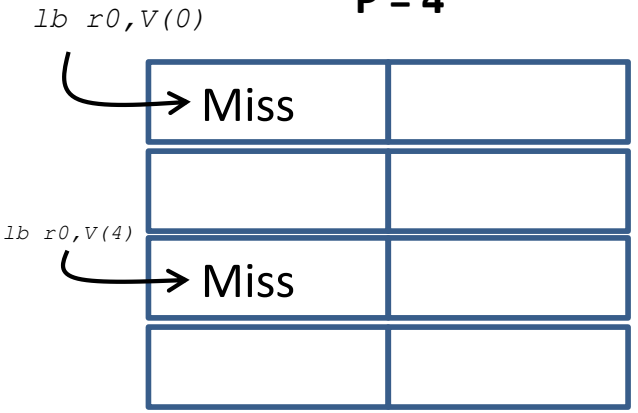
Some accesses hit in the cache → **Execution time lower than cases P=2,4**

P = 2



All accesses hit in the cache → **Execution time larger tan case P=1**

P = 4



All accesses hit in the cache → **Execution time similar to case P=2**

¿How do you discover block size?

Look at the same column of memory accesses and make sure that the cache is overflowing to ensure that at least capacity misses occur

Board: DE0-Nano

P			Number of memory accesses			E
Pattern	128	256	512	1024	2048	4096
1	20	40	85	163	320	946
2	20	46	85	163	633	1258
4	27	46	85	476	947	1888
8	24	43	239	480	945	1888

BLOCK: 4 B

Execution time

Ratio: 480/476 = 1,0

Ratio: 476/163 = 2,9