



PRESENTACION
FASE 1
EQUIPO NULL



CONTENIDOS

- 01 Algoritmo
- 02 Código Verilog
- 03 Reporte
- 04 Código python

SUMA DE DIGITOS

```
sum_of_digits:  
    addi $sp, $sp, -8    # allocate space for return  
                        address and sum  
  
    sw $ra, 4($sp)      # save return address  sw  
    $zero, 0($sp)       # initialize sum to 0  bgt $a0,  
    $zero, loop          # if n > 0 go to loop  
    lw $t0, 4($sp)       # load return address  
    lw $v0, 0($sp)       # load sum  
    addi $sp, $sp, 8     # deallocate stack space jr  
    $t0                 # return
```

```
loop:  
    div $a0, $a0, 10    # divide n by 10  
    mfhi $t1 # get remainder  
    add $v0, $v0, $t1    # add remainder to sum  
    bgt $a0, $zero, loop # if n > 0 go to loop  
  
done:  
    lw $t0, 4($sp)      # load return address  lw  
    $v0, 0($sp)          # load sum  
    addi $sp, $sp, 8     # deallocate stack space jr  
    $t0                 # return
```

CODIGO VERILOG

```
module singledata(
    input clk_tb,
    output ini_tb
);
wire [31:0] C1; // output PC
wire [31:0] C2; //output ADD
wire [31:0] C3; //output instruction_mempry
wire [31:0] C4; //WD y salida de data_memory
wire [31:0] C5; //op1
wire [31:0] C6; //op2
wire [3:0] C7; //Alu_ctrl
wire [31:0] C8; //res
wire [5:0] C9; //Aluop
wire [31:0] C10; //Mux
wire [31:0] C11; //Write
wire [31:0] C12; //Read
wire [31:0] C13; //input Mux
ADD singleA(
    .addr(C8),
    .write_data(C11),
    .write_enable(C12),
    .read_data(C13)
    .a(),
    .b(C1),
    .suma(C2)
);
PC singleB (
    .clk(clk_tb),
    .in(C2),
    .pc(C1)
);

instruction_memory
singleC (
    .address(C1),
    .instruction(C3)
);
bdr bdr_inst (
    .RegEn(C3[31:26]),
    .RR1(C3[25:21]),
    .RR2(C3[20:16]),
    .WA([C3[15:11]]),
    .WD(C4),
    .DR1(C5),
    .DR2(C6));
ALU ALU_inst (
    .op1(C5),
    .op2(C6),
    .alu_ctrl(C7),
    .res(C8),
    .zf()
);

alu_control
alu_control_inst(
    .opcode(C3),
    .ALUop(C9),
    .alu_ctrl(C7)
);

data_memory data_inst(
    .clk(),
    .addr(C8),
    .write_data(C11),
    .write_enable(C12),
    .read_data(C13)
);

control_unit control_unit_inst(
    .opcode(C3),
    .RegWrite(C10),
    .MemWrite(C11),
    .MemRead(C12),
    .Branch(),
    .ALUOp(C9)
);

mux mux_inst(
    .a(C13),
    .b(C8),
    .sel(C10),
    .y(C4)
);

control_unit control_unit_inst(
    .opcode(C3),
    .RegWrite(C10),
    .MemWrite(C11),
    .MemRead(C12),
    .Branch(),
    .ALUOp(C9)
);

mux mux_inst(
    .a(C13),
    .b(C8),
    .sel(C10),
    .y(C4)
);

//assign bdr_inst_addr = C1;
//assign singleC_address =
bdr_inst_addr;
//assign instruction = C3;
```

REPORT

El procesador MIPS de 32 bits es un procesador RISC que cuenta con una unidad de control, una ALU, 32 registros de propósito general, una memoria caché, una FPU y una arquitectura de pipeline. Su conjunto de instrucciones es relativamente pequeño y está diseñado para facilitar la programación y mejorar el rendimiento.

Ljubisa Bajic es un ingeniero en computación que ha trabajado en el diseño de microprocesadores durante más de 30 años.

Jim Keller es otro ingeniero en computación que ha sido fundamental en la evolución de los microprocesadores. Keller ha trabajado en el diseño de procesadores de alta gama durante más de 20 años

CODIGO PYTHON

PYTHON

```
def suma_digitos(n):
    if n < 10:
        return n
    else:
        return n % 10 + suma_digitos(n // 10)
```



FECHAS Y ASISTENCIAS

Primera reunion :

Jueves 27 de abril

Asistieron todos los integrantes

Segunda reunion:

Lunes 1 de mayo

Asistieron todos los integrantes

Tercera reunion:

Sabado 06 de mayo

Asistieron todos los integrantes .