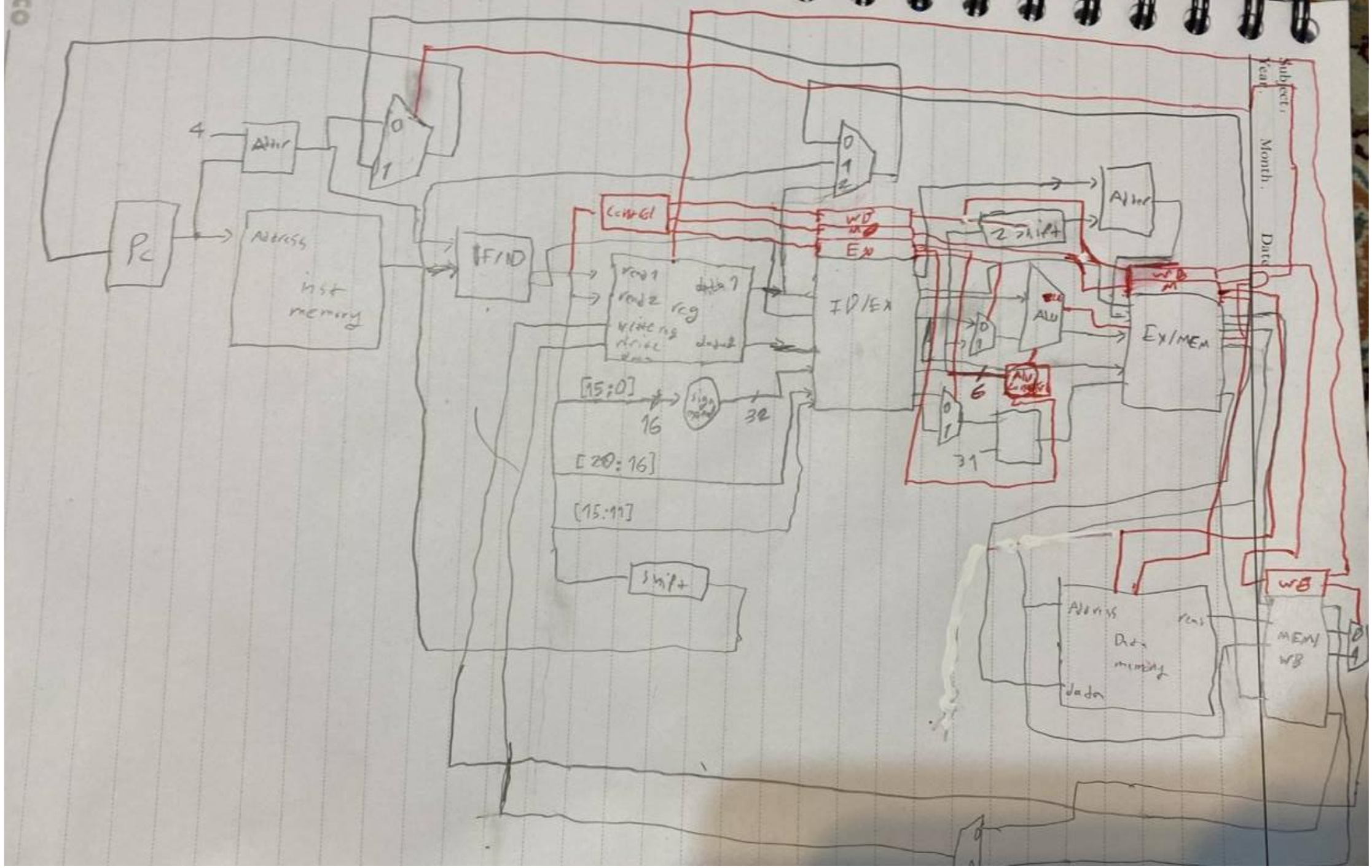
MIPS pipeline processor

Amir Hossein Shagholi 810199441
Erfan Soleymani 810199439


4
Adder
PC
Address
inst
memory
IF/ID
control
reg
IF/ID
ALU
Ex/MEM
[15:0]
16
32
[20:16]
[15:11]
shift
31
6
Address
Data
memory
Data
MEM/
WB
WB
Subject:
Month.
Date

```
        alu_src = 1'b0;
        mem_read= 1'b0;
        mem_write=1'b0;
        pc_src = 2'b00;
        alu_op = 2'b00;
        flush =1'b0;
        case (opcode)
            6'b000000 : {reg_dst, reg_write, alu_op} = {2'b01, 1'b1, 2'b10};
            6'b100011 : {alu_src, mem_to_reg, reg_write, mem_read} = {1'b1, 2'b01, 1'b1, 1'b1};
            6'b101011 : {alu_src, mem_write} = 2'b11;
            6'b000100 : {pc_src, flush} = {1'b0, operands_equal, operands_equal};
            6'b001001 : {reg_write, alu_src} = 2'b11;
            6'b000010 : {pc_src, flush} = {2'b10, 1'b1};
            6'b000011 : {reg_dst, mem_to_reg, pc_src} = {2'b10, 2'b10, 2'b10};
            6'b000110 : {pc_src} = {2'b11};
            6'b001010 : {alu_src, reg_dst, reg_write, alu_op, mem_to_reg} = {1'b1, 2'b00, 1'b1, 2'b11, 2'b00};
        endcase
    end
endmodule
```

Controller

inst.txt - Notepad

File Edit Format View Help

```
10001100000001000000001111101000
00100100000001010000000000000000
00100100000000010000000000000001
00100100000000010000000000010100
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00010000001000100000000000001110
00000000000000001000110000100000
10001100011001110000001111101000
00000000100001110011000000101010
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00010000110000000000000000000010
00000000000001110010000000100000
00000000000000001001010000100000
00100100001000010000000000000001
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00001000000000000000000000000111
10101100000001000000011111010000
10101100000001010000011111010100
```

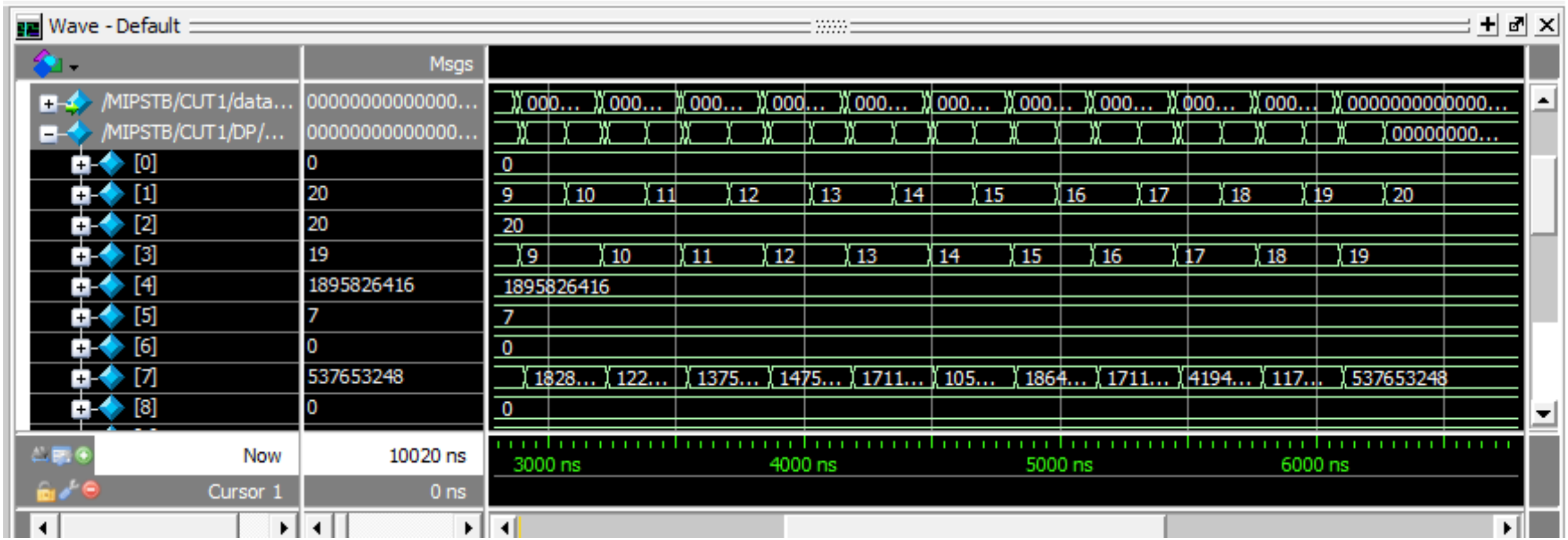
Inst codes

mem.txt - Notepad

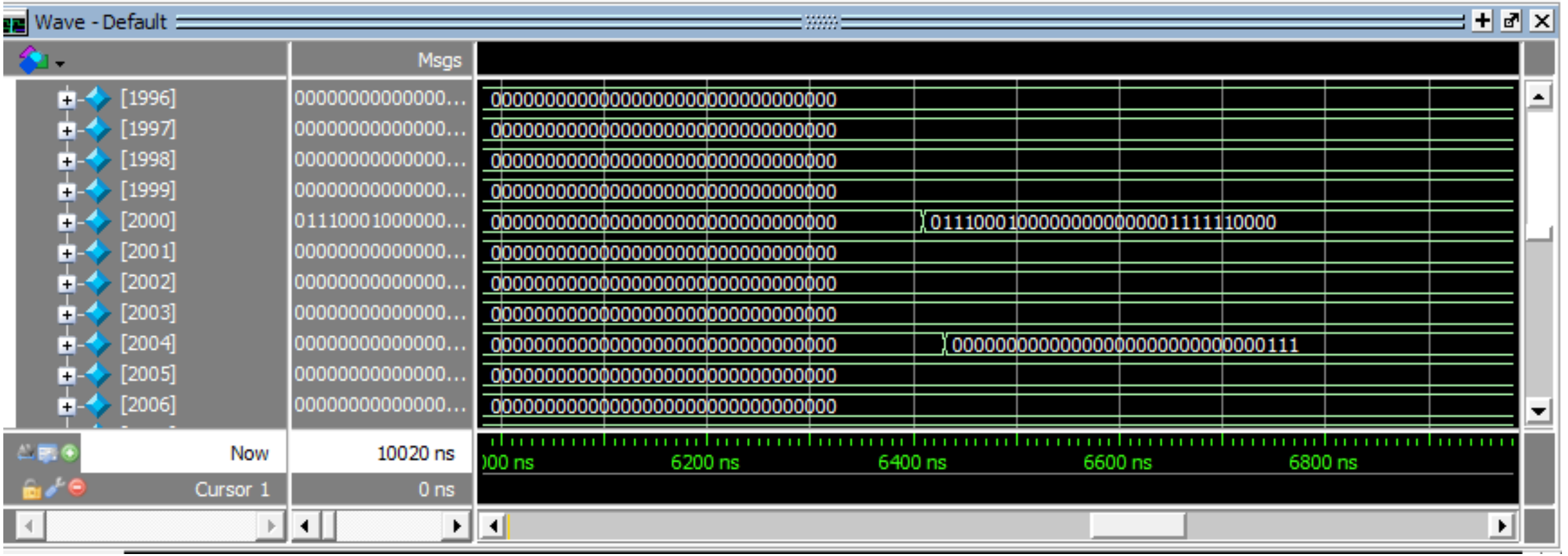
File Edit Format View Help

```
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00100101000001010100001001010000
00011011000000000001111110000000
00110001000101010101001100000000
01011110000011010100101111000000
00000001010101000100010001110001
00011100000000001100010100010000
01010011000001110000000000110000
01110001000000000000001111110000
00111110000010010000001101001000
01101101000000110000000001111000
01001001000110101000100010110000
01010010000000001010101000111000
01010111111110001111111111011111
01100110000000001000010010010011
00111111000000001000001000101000
01101111000111011110000000000000
01100101111111111111111101111111
00011001000000000000000000000000
01000101110100101010001111100000
00100000000001011111100000000000
00010100101011100100101010000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
```

Memory data [1000:1020]



Register[4] will be used to store the biggest data
Register[5] will be used to store the biggest data andis



Memory[2000] : biggest data (01110001000000000000001111110000)
Memory[2004] : biggest data andis (00000000000000000000000000000111)