

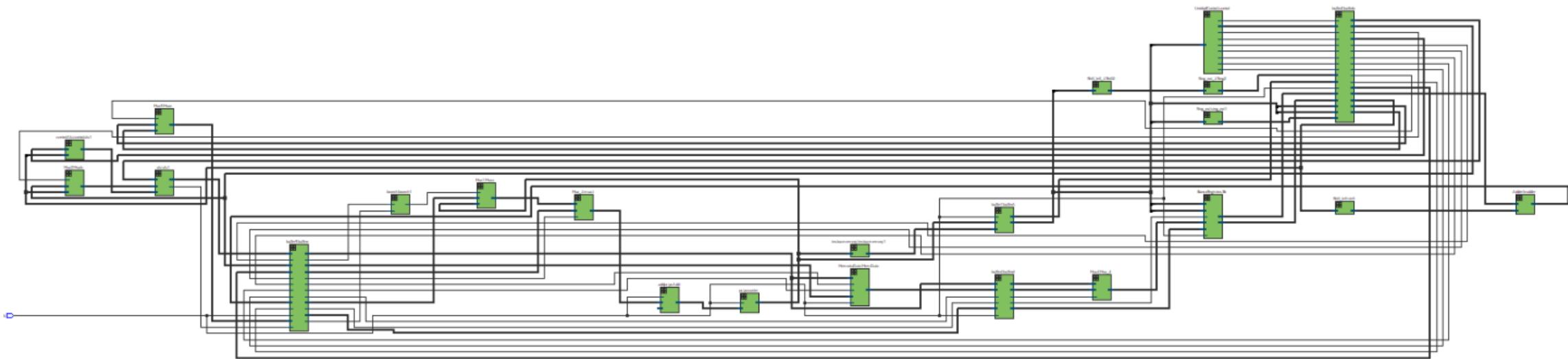
FASE 3 (PROYECTO MIPS)

GONZÁLEZ RAMOS JORGE HUMBERTO

OCHOA VELASCO DANA

JIMÉNEZ VITAL FRANCISCO

Datapath Fase 3



Resumen Fase 3

- ▶ Esta fase ya tiene la capacidad suficiente para ejecutar las instrucciones básicas de un procesador, se implementa la fase J sin problemas.

The timing diagram illustrates the memory access and control signals across four clock cycles. The left column lists the signal names and their initial values:

- Data1[31:0] = 0
- Data2[31:0] = 0
- Read1[4:0] = 0
- Read2[4:0] = 0
- RegWrite[0] = 1
- WriteDat[31:0] = 0
- WriteReg[4:0] = 0
- clk[0] = 0

MEM

The right side shows the signal waveforms for each cycle. The Address[31:0] signal is shown as 1330, XXX, 606, and 1. The DataRead[31:0] signal is shown as XXX. The MemRead[0] signal is high during the first two cycles. The MemWrite[0] signal is high during the third cycle. The WriteData[31:0] signal is shown as 1337, 1330, 606, and 1. The clk[0] signal is a square wave.

Decodificador

- ▶ Se hizo el selective_sort en ensamblador.

```
asm > asm selective_sortasm
 9   li $t8, 39
10  li $t1, 69
11
12  sw $t5, 0($zero)
13  sw $t4, 1($zero)
14  sw $t3, 2($zero)
15  sw $t2, 3($zero)
16  sw $t1, 4($zero)
17  sw $t6, 5($zero)
18  sw $t7, 6($zero)
19  sw $t8, 7($zero)
20  sw $t0, 8($zero)
21  li $t9, 8 # total de nums
22
23 .sort:
24  add $t0, $zero, $zero # i = 0
25
26 .Loop1:
27  addi $t1, $t9, -1 # n - 1
28  add $t2, $t0, $zero # index = i
29  addi $t3, $t0, 1 # j = i+1
30
31 .Loop2:
32  nop
33  lw $t4, 0($t3) # arr[j]
34  lw $t5, 0($t2) # arr[index]
35
36  # if arr[j] < arr[index]
37  blt $t4, $t5 .change
38  j .addj
39  .change:
40  add $t2, $zero, $t3 # index = j
41
42  .addj:
43  addi $t3, $t3, 1 # j = j +1
44  blt $t3, $t9, .loop2 # mientras j < n
45
46 .swap:
47  lw $t4, 0($t2) # arr[index]
48  lw $t5, 0($t0) # arr[i]
49
50  add $a1, $zero, $t4 # temp = arr[index]
51
52  sw $t5, 0($t2) # arr[index] = arr[i]
53  sw $a1, 0($t0) # arr[i] = temp
54
55 .add_i:
56  addi $t0, $t0 1 # i = i + 1
57  blt $t0, $t1 .loop1 # mientras i < n - 1
58
59 .end:
60  lw $t1, 0($zero)
61  lw $t1, 1($zero)
62  lw $t1, 2($zero)
63  lw $t1, 3($zero)
64  lw $t1, 4($zero)
65  lw $t1, 5($zero)
66  lw $t1, 6($zero)
67  lw $t1, 7($zero)
68  lw $t1, 8($zero)
```

Resultado ensamblador

Entrada

13	0	12	0	11	0	10	0	8	0	14	0	15	0	24	0	9	0	13	
0		85	0	33	0	94	0	55	0	105	0	58	0	43	0	39	0	69	0
0		13	0	12	0	11	0	10	0	8	0	14	0	15	0	24	0	9	0

Salida