Ali Padyav - 810199388
Shayan Kashefi - 810199585

CA3 – Multi Cycle MIPS

دستورات:

R_type:

[31:26] 000000	[25:21] R_s	[20:16] R_t	[15:11] R_d	[10:6] sham	[5:0] func

```
func:    000 -> and   001 -> or
         010 -> add   110 -> sub
         111 -> slt
```

slti:

[31:26] 001010	[25:21] R_s	[20:16] R_s	[15:0] immediate

addi:

[31:26] 001001	[25:21] R_s	[20:16] R_s	[15:0] immediate

lw:

[31:26] 000000	[25:21] R_a	[20:16] R_d	[15:0] add

sw:

[31:26] 100011	[25:21] R_a	[20:16] R_s	[15:0] add

j:

[31:26] 001010	[25:0] address

jr:

[31:26] 000110	[25:21] R_s	-	-

jal:

[31:26] 000010	[25:0] address

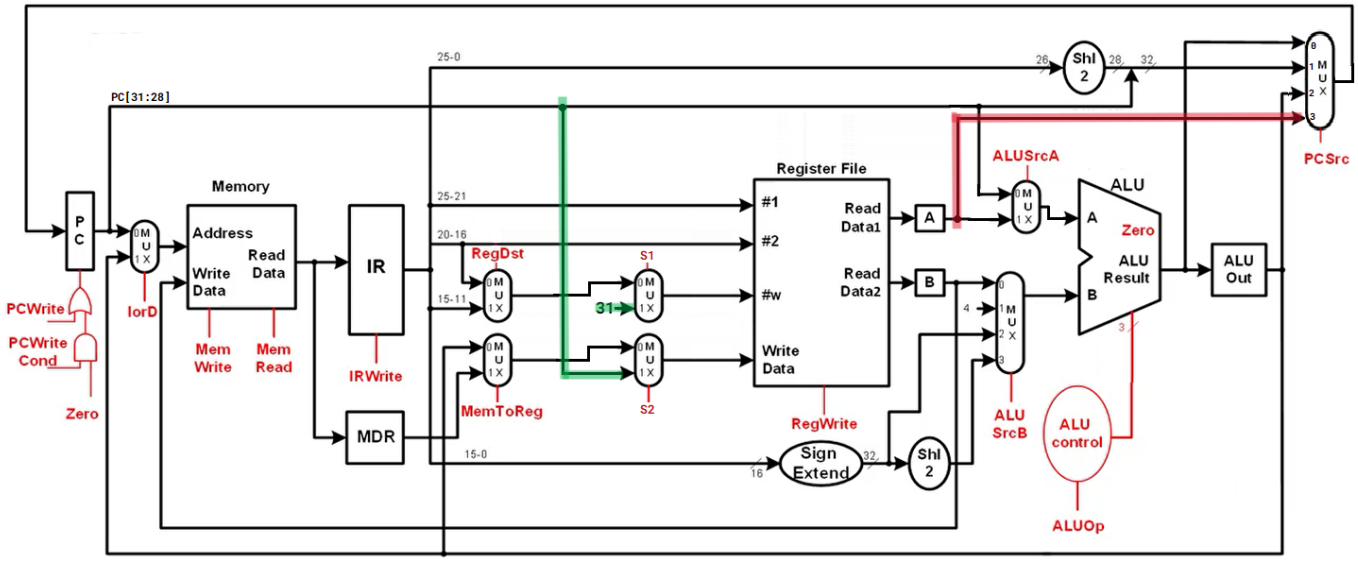
beq:

[31:26] 101011	[25:21] R_a	[20:16] R_b	[15:0] add

Ali Padyav - 810199388
Shayan Kashefi - 810199585

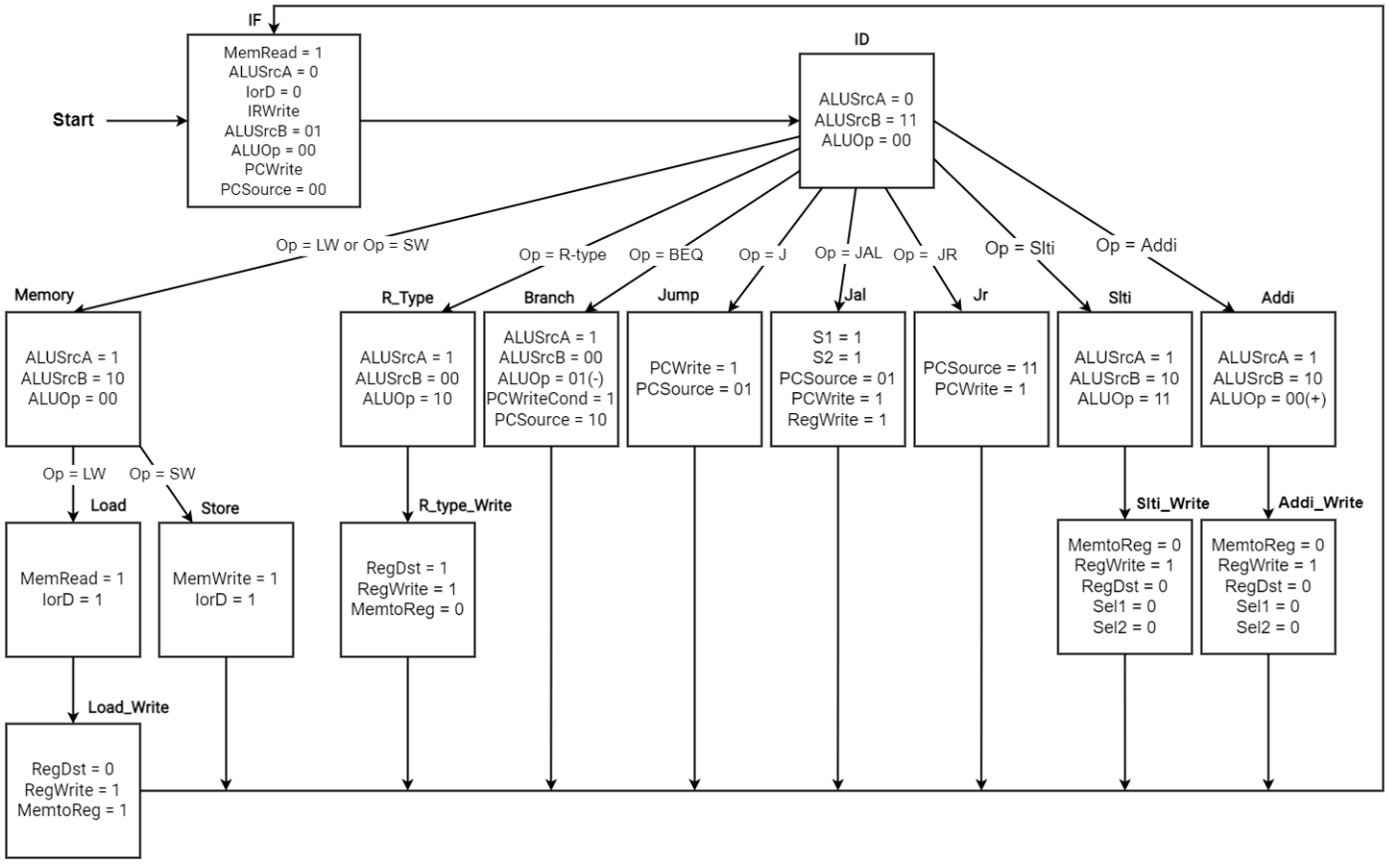
CA3 – Multi Cycle MIPS

Datapath:



رنگ سبز برای `jal` و رنگ قرمز برای `jr` اضافه شده است.

Controller:



Ali Padyav - 810199388
Shayan Kashefi - 810199585

CA3 – Multi Cycle MIPS

Assembly Program:

```
1  j main
2
3  find_min:
4      addi R1, R0, 1000              # i = 1000
5      lw   R2, 0(R1)                 # min = mem[1000]
6      addi R3, R0, 0                 # min_i = 0
7      addi R4, R0, 0                 # loop_count = 0
8
9      loop:
10         addi R1, R1, 4             # i += 4
11         addi R4, R4, 1             # loop += 1
12         slti R5, R4, 20            # check if loop_count passes 20
13         beq  R5, R0, end_loop      # if passes, jump to end_loop
14         lw   R6, 0(R1)             # new = mem[i]
15         slt  R5, R6, R2            # check if new(R6) is smaller than min(R2)
16         beq  R5, R0, loop          # if new is equal or greater than min, jump to loop. else go to next instruction
17         add  R2, R0, R6            # min = new
18         add  R3, R0, R4            # min_i = loop_count
19         j    loop                  # go to next element of array
20
21     end_loop:
22         sw R2, 2000(R0)            # save min in mem[2000]
23         sw R3, 2004(R0)            # save min_i in mem[2004]
24         jr R31                     # return to outer function
25
26
27 main:
28     jal find_min                   # call find_min function
```

Machine Code:

```
00010010    00000000    00000001    00000000    00100000    11010000    00000001
00000000    00000000    00000000    00000000    00010000    00000111    00000000
00000000    00000011    10000100    00100110    00000110    00000010    00000000
00001000    00100100    00100100    10001100    00000000    10101100    00001100

11101000    00000000    00010100    00101010    00100000    11010100
00000011    00000000    00000000    00101000    00011000    00000111
00000001    00000100    10000101    11000010    00000100    00000011
00100100    00100100    00101000    00000000    00000000    10101100

00000000    00000100    00000110    11111001    00000101    00000000
00000000    00000000    00000000    11111111    00000000    00000000
00100010    00100001    00000101    00000101    00000000    11100000
10001100    00100100    00010000    00010000    00001000    00011011
```

کد اسمبلی ما 19 خط بود. از آنجا که هر خانه حافظه 4 بایت است، تعداد خطوط instruction ما 76 خط شد.

Array:

آرایه 20 تایی که تست کردیم به شرح زیر است. مموری فایل این آرایه را به کمک Quartus ساختیم:

1	25
2	23
3	23
4	11
5	16
6	25
7	-3
8	6
9	-1
10	-56
11	89
12	12
13	-70
14	-128
15	56
16	56
17	-10
18	0
19	19
20	130

@3e8	@3fc	@410	@424
00011001	00011001	01011001	00111000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
@3ec	@400	@414	@428
00010111	11111101	00001100	11110110
00000000	11111111	00000000	11111111
00000000	11111111	00000000	11111111
00000000	11111111	00000000	11111111
@3f0	@404	@418	@42c
00010111	00000110	10111010	00000000
00000000	00000000	11111111	00000000
00000000	00000000	11111111	00000000
00000000	00000000	11111111	00000000
@3f4	@408	@41c	@430
00001011	11111111	10000000	00010011
00000000	11111111	11111111	00000000
00000000	11111111	11111111	00000000
00000000	11111111	11111111	00000000
@3f8	@40c	@420	@434
00010000	11001000	00111000	10000010
00000000	11111111	00000000	00000000
00000000	11111111	00000000	00000000
00000000	11111111	00000000	00000000

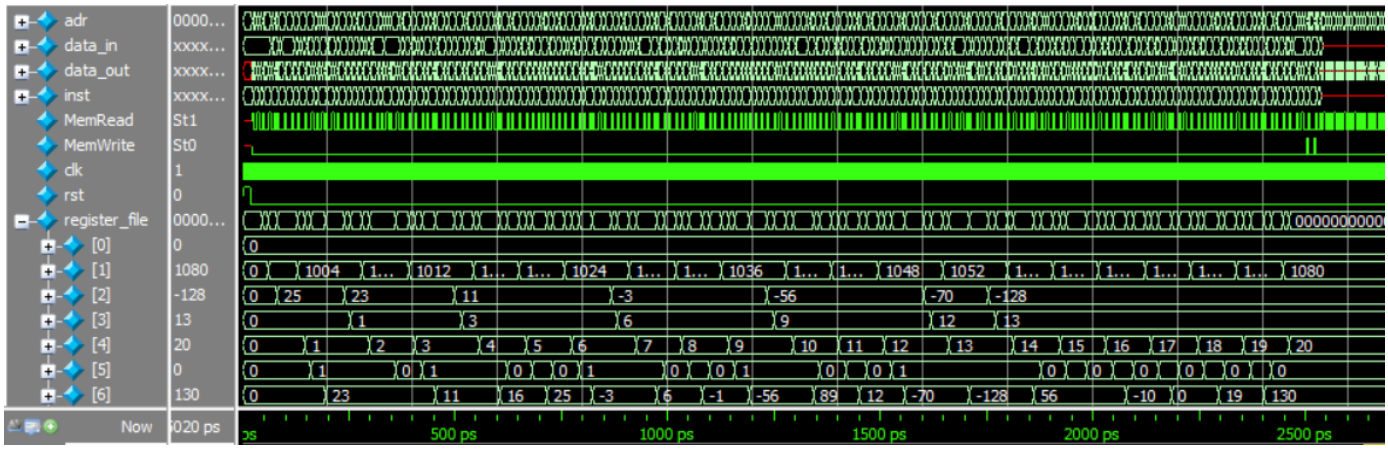
Code:

```
module data_mem (adr, d_in, mrd, mwr, clk, d_out);
  input [31:0] adr;
  input [31:0] d_in;
  input mrd, mwr, clk;
  output [31:0] d_out;

  reg [7:0] mem[0:65535];


  initial $readmemb("instructions.mem", mem);
  initial $readmemb("array.mem", mem, 1000);

  // The following initial block displays min elemnt in proper time
  initial
    #2600 $display("Min element is in mem[2000] = %d", $signed({mem[2003], mem[2002], mem[2001], mem[2000]}));

  always @(posedge clk)
    if (mwr==1'b1)
      {mem[adr+3], mem[adr+2], mem[adr+1], mem[adr]} = d_in;

  assign d_out = (mrd==1'b1) ? {mem[adr+3], mem[adr+2], mem[adr+1], mem[adr]} : 32'd0;

endmodule

```

مموری فایل دستورات و آرایه با دستور `$readmemb` در `data_mem` ذخیره میشود.

در زمان 2600 ps هم خانه 2000 حافظه display میشود که نشان دهنده کوچک ترین عنصر است.

Simulation:



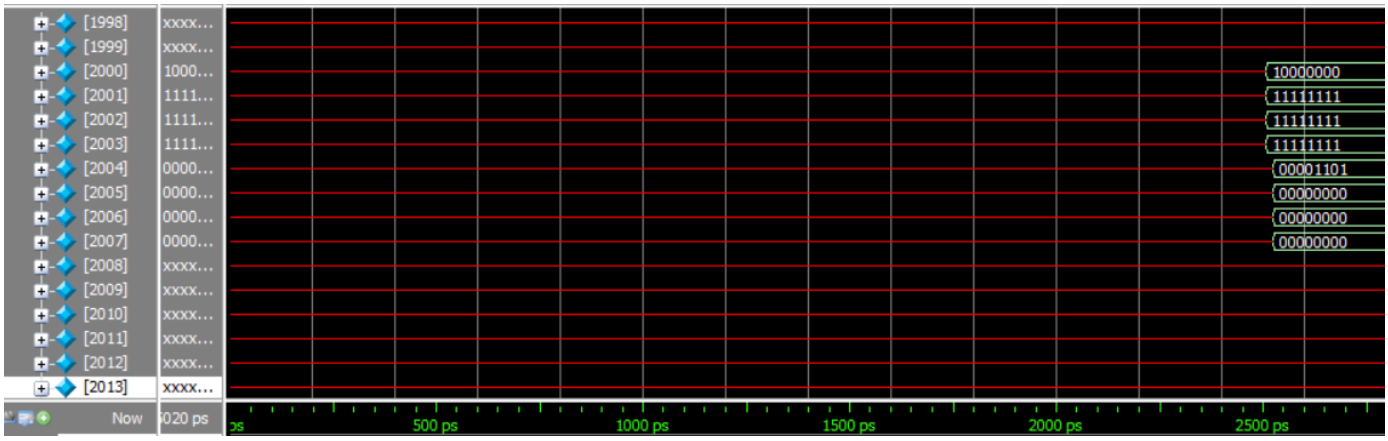
خانه 1 رجیستر نشان دهنده اندیس آرایه است.

خانه 2 رجیستر نشان دهنده کمترین عنصر پیدا شده است.

خانه 3 رجیستر نشان دهنده اندیس کمترین عنصر پیدا شده است.

خانه 4 رجیستر نشان دهنده حلقه هایی که تا الان زدیم است.

خانه 6 رجیستر نشان دهنده عنصر حال آرایه است.



مشخص است که مقدار عنصر و اندیسش در خانه 2000 و 2004 موری ذخیره شده است.