Ali Padyav - 810199388
Shayan Kashefi - 810199585

CA2 – Single Cycle MIPS

دستورات اضافه شده:

`slti:`

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

`j:`

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

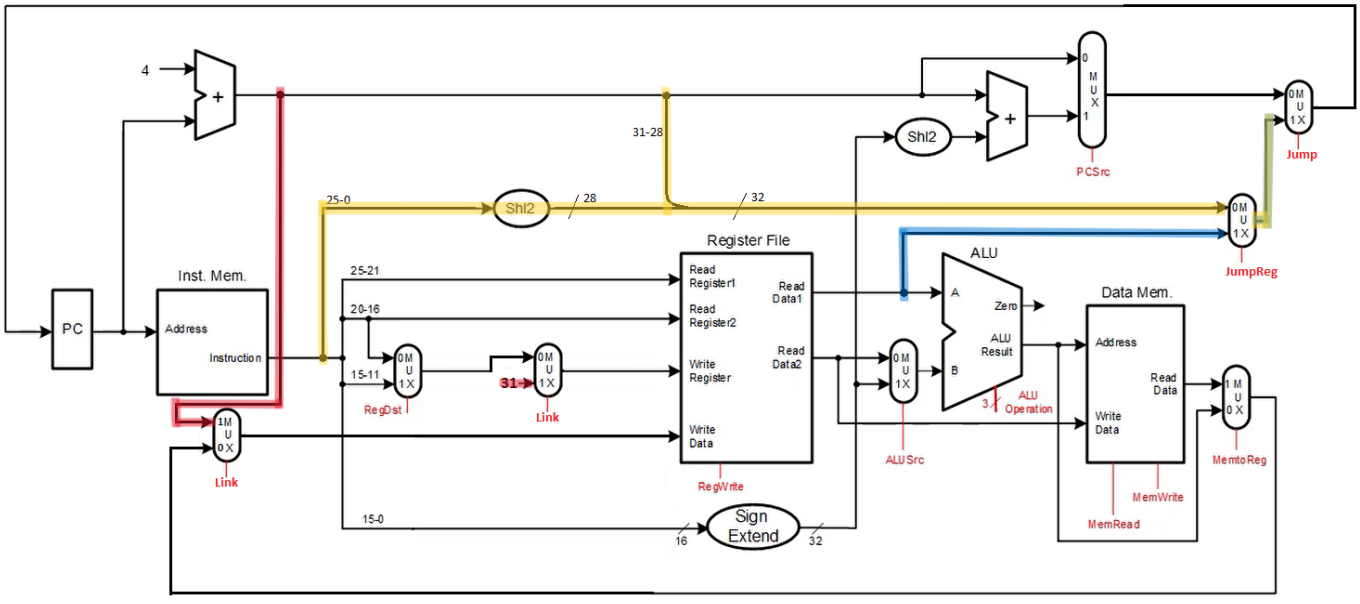
`jr:`

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

`Jal:`

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

Datapath:



رنگ قرمز برای jal، رنگ زرد برای j و رنگ آبی برای jr اضافه شده است.

Ali Padyav - 810199388
Shayan Kashefi - 810199585

CA2 – Single Cycle MIPS

Controller:

	regDst	ALUSrc	regWrite	memRead	memWrite	memToReg	ALUOp	PCSrc	Jump	JumpReg	Link
and	1	0	1	0	0	0	000(&)	0	0	-	0
or	1	0	1	0	0	0	001(|)	0	0	-	0
add	1	0	1	0	0	0	010(+)	0	0	-	0
sub	1	0	1	0	0	0	110(-)	0	0	-	0
slt	1	0	1	0	0	0	111(a<b)	0	0	-	0
addi	0	1	1	0	0	0	000(+)	0	0	-	0
lw	0	1	1	1	0	1	010(+)	0	0	-	0
sw	-	1	0	0	1	-	010(+)	0	0	-	0
beq	-	0	0	0	0	-	110(-)	zero	0	-	-
j	-	-	0	0	0	-	-	0	1	-	-
jr	-	-	0	0	0	-	-	-	1	1	-
jal	-	-	1	0	0	-	-	0	1	-	1
slti	0	1	1	0	0	0	000(+)	0	0	-	0

سیگنال‌های اضافه شده به کنترلر Jump، JumpReg و Link میباشند.

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	00000110	00100000	11010100
00000000	00000000	00000000	00010000	00000111
00000000	00000100	00000101	00000110	00000011
00001000	00100100	00010000	00000000	10101100
11101000	00000100	00000000	00100000	00000000
00000011	00000000	00000000	00011000	00000000
00000001	00100001	00100110	00000100	11100000
00100100	00100100	10001100	00000000	00011011
00000000	00000001	00101010	00000101	00000001
00000000	00000000	00101000	00000000	00000000
00100010	10000100	11000010	00000000	00000000
10001100	00100100	00000000	00001000	00001100
00000000	00010100	11111001	11010000	
00000000	00000000	11111111	00000111	
00000011	10000101	00000101	00000010	
00100100	00101000	00010000	10101100	

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

```
1  module inst_mem (adr, d_out);
2    input [31:0] adr;
3    output [31:0] d_out;
4
5    reg [7:0] mem[0:65535];
6
7    initial $readmemb("instructions.mem", mem);
8
9    assign d_out = {mem[adr[15:0]+3], mem[adr[15:0]+2], mem[adr[15:0]+1], mem[adr[15:0]]};
10 endmodule
```

بعد این instructions را با `$readmemb` از فایل میخوانیم و در `inst_mem` ذخیره میکنیم.

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

```
25
23
23
11
16
25
-3
6
-1
-56
89
12
-70
-128
56
56
-10
0
19
130
```

واضح است که در فایل مموری ساخته شده، آرایه از خانه 1000 حافظه شروع میشود.

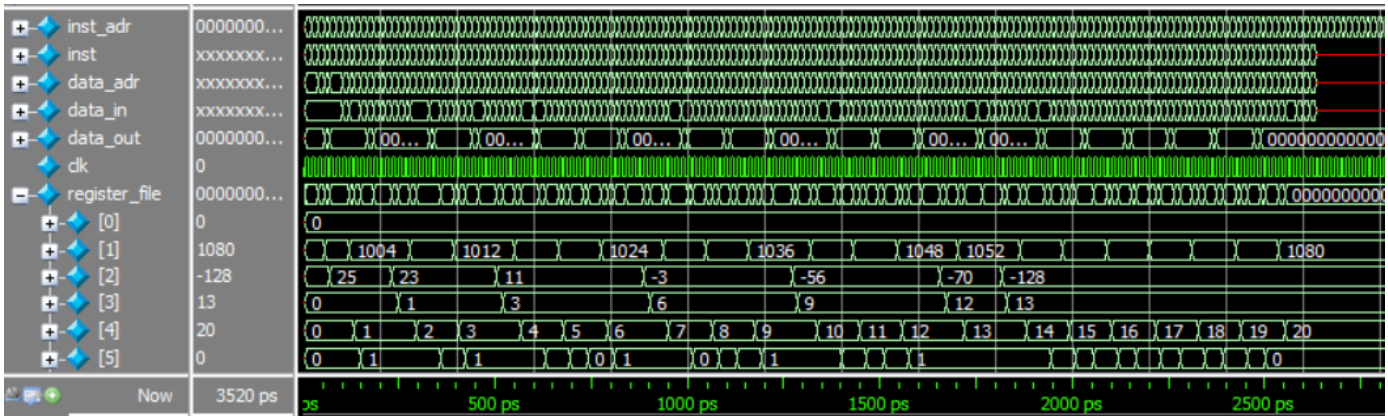
@3e8	@3f4	@400	@40c	@418	@424	@430
00011001	00001011	11111101	11001000	10111010	00111000	00010011
00000000	00000000	11111111	11111111	11111111	00000000	00000000
00000000	00000000	11111111	11111111	11111111	00000000	00000000
00000000	00000000	11111111	11111111	11111111	00000000	00000000
@3ec	@3f8	@404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

```
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("array.mem", mem, 1000);

  // The following initial block displays min elemnt in proper time
  initial
    #3300 $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 آرایه نیز با دستور $readmemb در data_mem ذخیره میشود.

در زمان 3300 ps هم خانه 2000 حافظه display میشود که نشان دهنده کوچک ترین عنصر است.

Simulation:

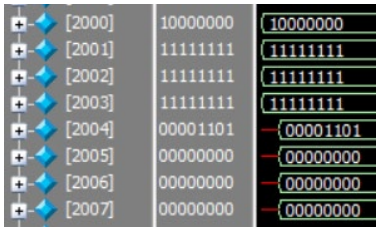


خانه 1 رجیستر نشان دهنده اندیس آرایه است.

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.



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