اعضای گروه:
سـوگل گـودرزی ۸۱۰۱۹۸۴۶۷
عرفـان پـنـــاهی ۸۱۰۱۹۸۳۶۹

❊ مسیر داده:

بخش اعظم مسیر داده ای که برای پردازنده مطلوب داریم مشابه مسیر داده پردازنده MIPS است با این تفاوت که برای چهار دستور جدیدی که اضافه شده است مسیر داده را به شکلی که در تصویر مشاهده می شود تغییر داده ایم. برای دستورات مذکور، مسیر داده به شکل زیر تغییر می کند:

❊ دستور های addi و slti:

برای این دو دستور هیچ عنصر جدیدی به مسیر داده اضافه نشده است و این دستورات با همان مسیر داده پردازنده MIPS قابل پیاده سازی هستند. تنها کافیست مقادیر سیگنال های کنترلی عناصر مسیر داده را به درستی مقداردهی کنیم که در بخش کنترلر، جدول سیگنال های کنترلی را برای این دستورات مشاهده می کنید.

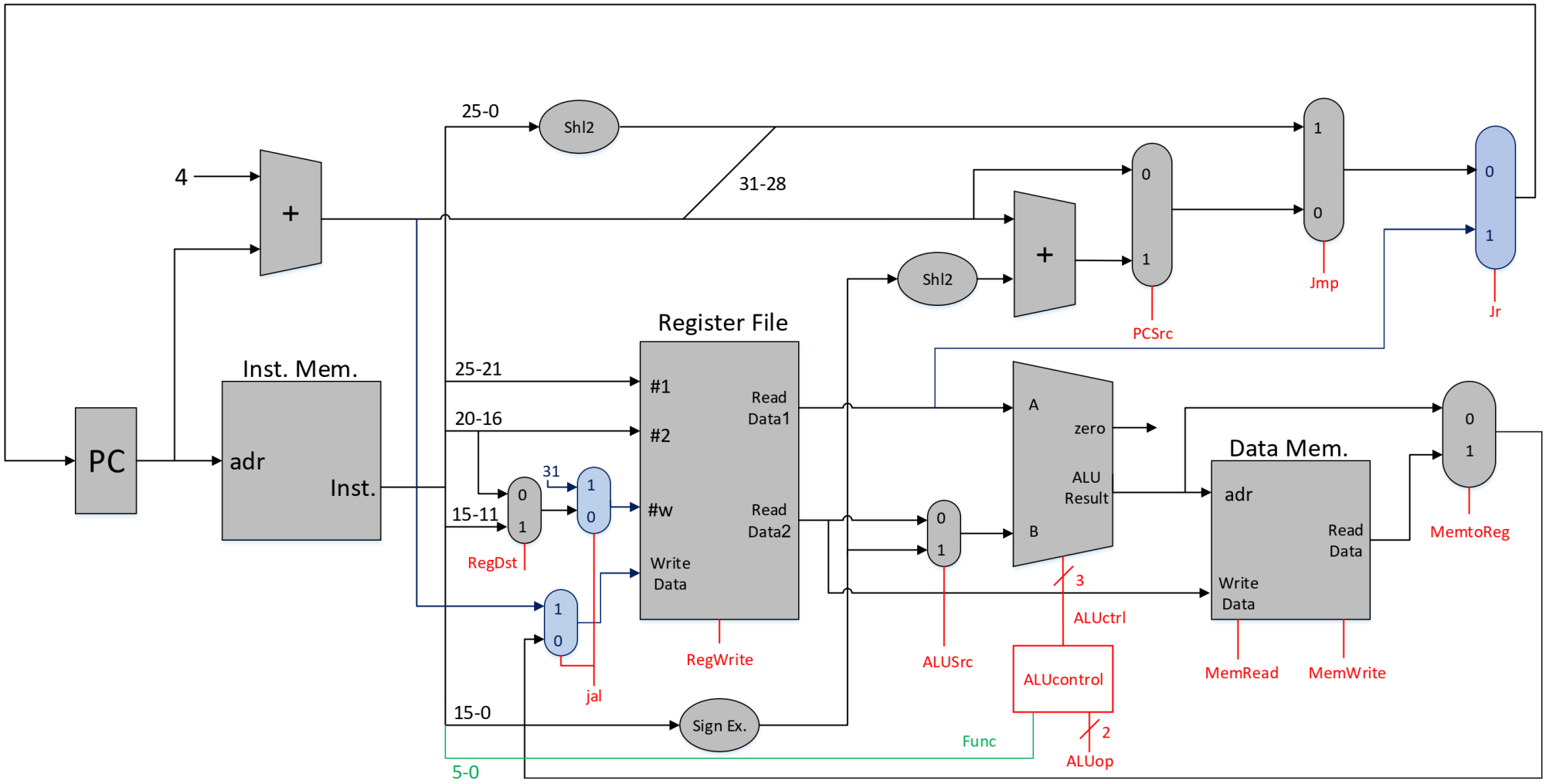
❊ دستور jal :

برای این دستور دو مالتیپلکسر برای ورودی های WriteRegister و WriteData در Register File اضافه کرده ایم. سیگنال های کنترلی این دو مالتیپلکسر یکسان هستند. در اینصورت وقتی jal مقدار یک داشته باشد این دستور انجام می شود و در غیر اینصورت بقیه دستورات بسته به حالت سیگنال های کنترلی دیگر انجام خواهند شد.

❊ دستور jr :

برای این دستور یک مالتیپلکسر قرار داده ایم که خروجی آن به PC متصل شده است چون با انجام شدن این دستور قرار است PC به آدرسی که در رجیستر مدنظر قرار دارد، منتقل شود. سیگنال کنترلی این مالتیپلکسر jr است که در صورتی که مقدار یک داشته باشد، این دستور انجام می شود و در غیر اینصورت دستور انجام نمی شود.

در صفحه بعد تغییرات اسجاد شده در مسیر داده با رنگ **آبی** مشخص شده است:


4
+
25-0
Shl2
31-28
Shl2
+
0
1
PCSrc
1
0
Jmp
0
1
Jr
PC
Inst. Mem.
adr
Inst.
25-21
20-16
15-11
RegDst
31
1
0
jal
1
0
Register File
#1
#2
#w
Write Data
Read Data1
Read Data2
RegWrite
0
1
ALUSrc
A
B
zero
ALU Result
3
ALUctrl
ALUcontrol
2
ALUop
Data Mem.
adr
Write Data
Read Data
MemRead
MemWrite
0
1
MemtoReg
15-0
Sign Ex.
Func
5-0

* کنترلر:

ابتدا ALUop را طوری تعیین می‌کنیم تا بتواند با استفاده از Func ، سلکتور های ALU (ALUctrl) را بسازد:

OPC	ALUop	Func	ALU Operation
000000	00	000001	Add → 000
000000	00	000010	Sub → 001
000000	00	000100	And → 010
000000	00	001000	Or → 011
000000	00	010000	Slt → 100
000001(addi)	01	-	Add → 000
000010(slti)	10	-	Slt → 100
000011(lw)	01	-	Add → 000
000100(sw)	01	-	Add → 000
000101(beq)	11	-	Sub → 001

حال برای قسمت کنترلر ، سیگنال های کنترلی را طبق جدول زیر مقداردهی می کنیم:

	RegDst	jal	RegWrite	ALUSrc	ALUop	MemRead	MemWrite	MemtoReg	PCSrc	jmp	jr
R-T	1	0	1	0	00	0	0	0	0	0	0
addi	0	0	1	1	01	0	0	0	0	0	0
slti	0	0	1	1	10	0	0	0	0	0	0
lw	0	0	1	1	01	1	0	1	0	0	0
sw	0	0	0	1	01	0	1	0	0	0	0
j	0	0	0	0	00	0	0	0	0	1	0
jal	0	1	1	0	00	0	0	0	0	1	0
jr	0	0	0	0	00	0	0	0	0	0	1
beq	0	0	0	0	11	0	0	0	zero	0	0

*** الگوریتم پیدا کردن بزرگترین مقدار یک آرایه ۲۰ عنصری و اندیس آن:**

```
array A[0:19];
Value = A[0];
Index =0;
for (int i=1;i<20;i++){
        if(Value < A[i]){
                Value = A[i];
                Index = i;
        }
}
//Value: Maximum value of A
//Index: Index of Maximum value of A
```

❊ برنامه با زبان اسمبلی و زبان ماشین:

برای انجام دستورات مناسب به منظور اینکه بزرگترین مقدار یک آرایه ۲۰ عنصری و همچنین اندیس آن را بیابیم و در آدرس های ۲۰۰۰ و ۲۰۰۴ حافظه بنویسیم، دستورات زیر را ابتدا به زبان کد ماشین نوشته سپس معادل باینری آن ها را پیدا کرده ایم و در آخر نیز آن ها را به اعداد هگزادسیمال تبدیل کرده و در فایل Instructions.txt ذخیره سازی می کنیم:

❊ اسمبلی:

```
        addi R30,R0,0
        addi R20,R0,0
        lw R1,R0(1000)
        addi R2,R0,0
        jal Loop //101
        sw R1,R0(2000)
        sw R2,R0(2004)

endL:   jr R31
Loop:   addi R20,R20,1
        slti R10,R20,19
        addi R11,R10,403
        addi R30,R30,4
        lw R3,R30(1000)
        slt R4,R1,R3
        beq R4,R0,L //2
        add R1,R3,R0
        add R2,R20,R0
G:      jr R11
```

❊ ماشین باینری:

0. 0000-0100-0001-1110-0000-0000-0000-0000
1. 0000-0100-0001-0100-0000-0000-0000-0000
2. 0000-1100-0000-0001-0000-0011-1110-1000
3. 0000-0100-0000-0010-0000-0000-0000-0000
4. 0010-0000-0000-0000-0000-0000-0110-0101
5. 0001-0000-0000-0001-0000-0111-1101-0000
6. 0001-0000-0000-0010-0000-0111-1101-0100
.

.

100. 0001-1111-1110-0000-0000-0000-0000-0000
101. 0000-0110-1001-0100-0000-0000-0000-0001
102. 0000-1010-1000-1010-0000-0000-0001-0011
103. 0000-0101-0100-1011-0000-0001-1001-0011
104. 0000-0111-1101-1110-0000-0000-0000-0100
105. 0000-1111-1100-0011-0000-0011-1110-1000
106. 0000-0000-0010-0011-0010-0000-0001-0000
107. 0001-0100-0000-0100-0000-0000-0000-0010
108. 0000-0000-0110-0000-0000-1000-0000-0001
109. 0000-0010-1000-0000-0001-0000-0000-0001
110. 0001-1101-0110-0000-0000-0000-0000-0000

❊ ماشین هگزادسیمال:

0. 041e0000

1. 04140000

2. 0c0103e8

3. 04020000

4. 20000065

5. 100107d0

6. 100207d4

.

100. 1fe00000

101. 06940001

102. 0a8a0013

103. 054b0193

104. 07de0004

105. 0fc303e8

106. 00232010

107. 14040002

108. 00600801

109. 02801001

110. 1d600000

* داده های تست:

درنهایت نیز داده هایی بعنوان مثال مانند تصویر زیر به برنامه میدهیم و خروجی را مشاهده میکنیم که نشان میدهد برنامه به درستی کار میکند:

249	xxxxxxxx
250	xxxxxxxx
251	00000001
252	00000002
253	00000003
254	00000004
255	00000005
256	00000006
257	00000007
258	00000008
259	00000009
260	0000000a
261	0000000b
262	0000000c
263	0000000d
264	0000000e
265	0000000f
266	00000020
267	00000011
268	00000012
269	00000013
270	00000014

آدرس های ۱۰۰۰ تا ۱۰۷۶ (داده ها)

500	xxxxxxxx
501	xxxxxxxx
502	xxxxxxxx
503	xxxxxxxx
504	00000020
505	0000000f
506	xxxxxxxx
507	xxxxxxxx
508	xxxxxxxx
509	xxxxxxxx
510	xxxxxxxx
511	xxxxxxxx
512	xxxxxxxx
513	xxxxxxxx
514	xxxxxxxx
515	xxxxxxxx
516	xxxxxxxx
517	xxxxxxxx
518	xxxxxxxx
519	xxxxxxxx
520	xxxxxxxx
521	xxxxxxxx

آدرس های ۲۰۰۰ و ۲۰۰۴ (نتیجه)