اعضای گروه:
سـوگل گـودرزی ۸۱۰۱۹۸۴۶۷
عرفـان پـنـــاهی ۸۱۰۱۹۸۳۶۹

* مسیر داده:

طبق مطالبی که در درس آموختیم برای پیاده سازی مسیر داده پردازنده به صورت پایپ لاین نیاز است که ساختار مسیر داده پردازنده تک مرحله ای را به گونه ای تغییر دهیم تا ساختار مربوط به هر مرحله (stage) در مسیر داده پایپ لاین مشخص شود. در این حالت باید بین stage ها از رجیسترهایی استفاده کنیم که این رجیسترها به منظور انتقال مناسب داده ها بین stage ها قرار می گیرند. پیاده سازی توضیحات گفته شده، در مسیر داده ای که در فایل ضمیمه شده است، قرار دارد.

* 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	EQ	0	0

*** الگوریتم پیدا کردن کوچکترین 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: Minimum value of A
//Index: Index of Minimum value of A
```

* برنامه با زبان اسمبلی و زبان ماشین (باینری و هگزادسیمال):

برای انجام دستورات مناسب به منظور اینکه کوچکترین 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:

Address	Instruction	Binary	Hexadecimal
0	addi R10,R0,0	0000-0100-0000-1010-0000-0000-0000-0000	040a0000
1	addi R2,R0,0	0000-0100-0000-0010-0000-0000-0000-0000	04020000
2	addi R20,R0,0	0000-0100-0001-0100-0000-0000-0000-0000	04140000
3	addi R30,R0,1	0000-0100-0001-1110-0000-0000-0000-0001	041e0001
4	addi R5,R0,1	0000-0100-0000-0101-0000-0000-0000-0001	04050001
5	lw R1,R10(1000)	0000-1101-0100-0001-0000-0011-1110-1000	0d4103e8
6	Jal 400 //Loop	0010-0000-0000-0000-0000-0000-0110-0100	20000064
7	NOP(add R0,R0,R0)	0000-0000-0000-0000-0000-0000-0000-0001	00000001
8	sw R1,R0(2000)	0001-0000-0000-0001-0000-0111-1101-0000	100107d0
9	sw R2,R0(2004)	0001-0000-0000-0010-0000-0111-1101-0100	100207d4
...	...	...	...
100	beq R30,R5,3	0001-0111-1100-0101-0000-0000-0000-0011	17c50003
101	NOP(add R0,R0,R0)	0000-0000-0000-0000-0000-0000-0000-0001	00000001
102	Jr R31	0001-1111-1110-0000-0000-0000-0000-0000	1fe00000

103	NOP(add R0,R0,R0)	0000-0000-0000-0000-0000-0000-0000-0001	00000001
104	addi R10,R10,4	0000-0101-0100-1010-0000-0000-0000-0100	054a0004
105	addi R20,R20,1	0000-0110-1001-0100-0000-0000-0000-0001	06940001
106	lw R3,R10(1000)	0000-1101-0100-0011-0000-0011-1110-1000	0d4303e8
107	slt R4,R1,R3	0000-0000-0010-0011-0010-0000-0001-0000	00232010
108	slti R5,R20,19	0000-1010-1000-0101-0000-0000-0001-0011	0a850013
109	addi R25,R0,400	0000-0100-0001-1001-0000-0001-1001-0000	04190190
110	beq R4,R30,3	0001-0111-1100-0100-0000-0000-0000-0011	17c40003
111	NOP(add R0,R0,R0)	0000-0000-0000-0000-0000-0000-0000-0001	00000001
112	add R1,R3,R0	0000-0000-0110-0000-0000-1000-0000-0001	00600801
113	add R2,R20,R0	0000-0010-1000-0000-0001-0000-0000-0001	02801001
114	Jr R25	0001-1111-0010-0000-0000-0000-0000-0000	1f200000
115	NOP(add R0,R0,R0)	0000-0000-0000-0000-0000-0000-0000-0001	00000001

* داده های تست:

درنهایت نیز داده هایی بعنوان مثال مانند تصویر زیر به برنامه می دهیم و خروجی را مشاهده می کنیم که نشان می دهد برنامه به درستی کار می کند:

change.log | Data.txt

```
246 xxxxxxxx
247 xxxxxxxx
248 xxxxxxxx
249 xxxxxxxx
250 xxxxxxxx
251 00000107
252 00000040
253 00000002
254 00000004
255 00000005
256 00000006
257 0000000c
258 00000008
259 00000009
260 0000000a
261 0000100b
262 00000003
263 0001000d
264 0000000e
265 0000000f
266 00000020
267 00000011
268 00000012
269 01000013
270 00000015
271
```

آدرس های ۱۰۰۰ تا ۱۰۷۶ (داده ها)

NewData.txt

```
493 xxxxxxxx
494 xxxxxxxx
495 xxxxxxxx
496 xxxxxxxx
497 xxxxxxxx
498 xxxxxxxx
499 xxxxxxxx
500 xxxxxxxx
501 xxxxxxxx
502 xxxxxxxx
503 xxxxxxxx
504 00000002
505 00000002
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
```

آدرس های ۲۰۰۰ و ۲۰۰۴ (نتیجه)

change.log | Data.txt

```
244 xxxxxxxx
245 xxxxxxxx
246 xxxxxxxx
247 xxxxxxxx
248 xxxxxxxx
249 xxxxxxxx
250 xxxxxxxx
251 00000107
252 00000040
253 00000003
254 00000004
255 90000005
256 00000006
257 00000001
258 00000008
259 00000009
260 0000000a
261 0000000b
262 0000000c
263 0001000d
264 0000000e
265 0000000f
266 00000020
267 80000011
268 00000012
269 01000013
270 00000015
271
```

آدرس های ۱۰۰۰ تا ۱۰۷۶ (داده ها)

NewData.txt

```
488 xxxxxxxx
489 xxxxxxxx
490 xxxxxxxx
491 xxxxxxxx
492 xxxxxxxx
493 xxxxxxxx
494 xxxxxxxx
495 xxxxxxxx
496 xxxxxxxx
497 xxxxxxxx
498 xxxxxxxx
499 xxxxxxxx
500 xxxxxxxx
501 xxxxxxxx
502 xxxxxxxx
503 xxxxxxxx
504 80000011
505 00000010
506 xxxxxxxx
507 xxxxxxxx
508 xxxxxxxx
509 xxxxxxxx
510 xxxxxxxx
511 xxxxxxxx
512 xxxxxxxx
513 xxxxxxxx
514 xxxxxxxx
515 xxxxxxxx
```

آدرس های ۲۰۰۰ و ۲۰۰۴ (نتیجه)

* طراحی پردازنده با سیگنال کنترلی برای فلاش کردن:

در طراحی قبلی برای اینکه bubble هایی که در درس توضیح داده شدند را به منظور برطرف سازی مخاطره های کنترلی اضافه کنیم، نیاز بود که دستورهای NOP را بعد از دستورات پرشی (beq , j , jr , jal) قرار دهیم. حال یک سیگنال کنترلی به نام IF_Flush به پردازنده اضافه می کنیم که این امر را برای ما انجام دهد. یعنی هر گاه در مسیر برنامه مدنظر، دستور پرشی تشخیص داده شود و حتما نیاز باشد پرش به آدرس مطلوب انجام شود، آن گاه این سیگنال توسط کنترلر set می شود تا bubble های مورد نیاز که همان دستورات NOP هستند، بعد از این دستورات پرشی به صورت خودکار توسط خود پردازنده تولید شوند. این سیگنال کنترلی همانطور ک از شکل مسیر داده مشخص است به رجیستر بین دو مرحله IF و ID متصل است و هرگاه مقدار آن یک شود، دستور NOP تولید شده و به مرحله ID می رود و بعد از دیکود شدن در این مرحله، به تک تک مراحل بعدی نیز منتقل می شود و مسیر اجرای برنامه به درستی صورت می گیرد.

* کنترلر:

جدول مربوط به کنترلر برنامه در حالت جدید با قرار دادن سیگنال IF_Flush به صورت زیر تغییر می کند:

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

* برنامه با زبان اسمبلی و زبان ماشین (باینری و هگزادسیمال):

در این حالت دیگر نیاز به اضافه کردن دستورات NOP به فایل Instructions نیست و این دستورات توسط خود پردازنده در صورت نیاز اضافه می شوند. پس دستورات الگوریتم برنامه به صورت زیر تغییر می کنند:

Address	Instruction	Binary	Hexadecimal
0	addi R10,R0,0	0000-0100-0000-1010-0000-0000-0000-0000	040a0000
1	addi R2,R0,0	0000-0100-0000-0010-0000-0000-0000-0000	04020000
2	addi R20,R0,0	0000-0100-0001-0100-0000-0000-0000-0000	04140000
3	addi R30,R0,1	0000-0100-0001-1110-0000-0000-0000-0001	041e0001
4	addi R5,R0,1	0000-0100-0000-0101-0000-0000-0000-0001	04050001
5	lw R1,R10(1000)	0000-1101-0100-0001-0000-0011-1110-1000	0d4103e8
6	Jal 400 //Loop	0010-0000-0000-0000-0000-0000-0110-0100	20000064
7	sw R1,R0(2000)	0001-0000-0000-0001-0000-0111-1101-0000	100107d0
8	sw R2,R0(2004)	0001-0000-0000-0010-0000-0111-1101-0100	100207d4
…	…	…	…
100	beq R30,R5,3	0001-0111-1100-0101-0000-0000-0000-0011	17c50003
101	Jr R31	0001-1111-1110-0000-0000-0000-0000-0000	1fe00000
102	addi R10,R10,4	0000-0101-0100-1010-0000-0000-0000-0100	054a0004
103	addi R20,R20,1	0000-0110-1001-0100-0000-0000-0000-0001	06940001

104	lw R3,R10(1000)	0000-1101-0100-0011-0000-0011-1110-1000	0d4303e8
105	slt R4,R1,R3	0000-0000-0010-0011-0010-0000-0001-0000	00232010
106	slti R5,R20,19	0000-1010-1000-0101-0000-0000-0001-0011	0a850013
107	addi R25,R0,400	0000-0100-0001-1001-0000-0001-1001-0000	04190190
108	beq R4,R30,3	0001-0111-1100-0100-0000-0000-0000-0011	17c40003
109	add R1,R3,R0	0000-0000-0110-0000-0000-1000-0000-0001	00600801
110	add R2,R20,R0	0000-0010-1000-0000-0001-0000-0000-0001	02801001
111	Jr R25	0001-1111-0010-0000-0000-0000-0000-0000	1f200000

* داده های تست:

درنهایت نیز همان داده هایی را که در برنامه قبلی برای سنجش صحت کارکرد برنامه داده بودیم را در اینجا به پردازنده می دهیم و مشاهده می کنیم که برنامه به درستی کار می کند:

change.log | Data.txt

```
246 xxxxxxxx
247 xxxxxxxx
248 xxxxxxxx
249 xxxxxxxx
250 xxxxxxxx
251 00000107
252 00000040
253 00000002
254 00000004
255 00000005
256 00000006
257 0000000c
258 00000008
259 00000009
260 0000000a
261 0000100b
262 00000003
263 0001000d
264 0000000e
265 0000000f
266 00000020
267 00000011
268 00000012
269 01000013
270 00000015
271
```

آدرس های ۱۰۰۰ تا ۱۰۷۶ (داده ها)

NewData.txt

```
493 xxxxxxxx
494 xxxxxxxx
495 xxxxxxxx
496 xxxxxxxx
497 xxxxxxxx
498 xxxxxxxx
499 xxxxxxxx
500 xxxxxxxx
501 xxxxxxxx
502 xxxxxxxx
503 xxxxxxxx
504 00000002
505 00000002
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
```

آدرس های ۲۰۰۰ و ۲۰۰۴ (نتیجه)

NewData.txt

```
488 xxxxxxxx
489 xxxxxxxx
490 xxxxxxxx
491 xxxxxxxx
492 xxxxxxxx
493 xxxxxxxx
494 xxxxxxxx
495 xxxxxxxx
496 xxxxxxxx
497 xxxxxxxx
498 xxxxxxxx
499 xxxxxxxx
500 xxxxxxxx
501 xxxxxxxx
502 xxxxxxxx
503 xxxxxxxx
504 80000011
505 00000010
506 xxxxxxxx
507 xxxxxxxx
508 xxxxxxxx
509 xxxxxxxx
510 xxxxxxxx
511 xxxxxxxx
512 xxxxxxxx
513 xxxxxxxx
514 xxxxxxxx
515 xxxxxxxx
```

آدرس های ۲۰۰۰ و ۲۰۰۴ (نتیجه)

change.log | Data.txt

```
244 xxxxxxxx
245 xxxxxxxx
246 xxxxxxxx
247 xxxxxxxx
248 xxxxxxxx
249 xxxxxxxx
250 xxxxxxxx
251 00000107
252 00000040
253 00000003
254 00000004
255 90000005
256 00000006
257 00000001
258 00000008
259 00000009
260 0000000a
261 0000000b
262 0000000c
263 0001000d
264 0000000e
265 0000000f
266 00000020
267 80000011
268 00000012
269 01000013
270 00000015
271
```

آدرس های ۱۰۰۰ تا ۱۰۷۶ (داده ها)