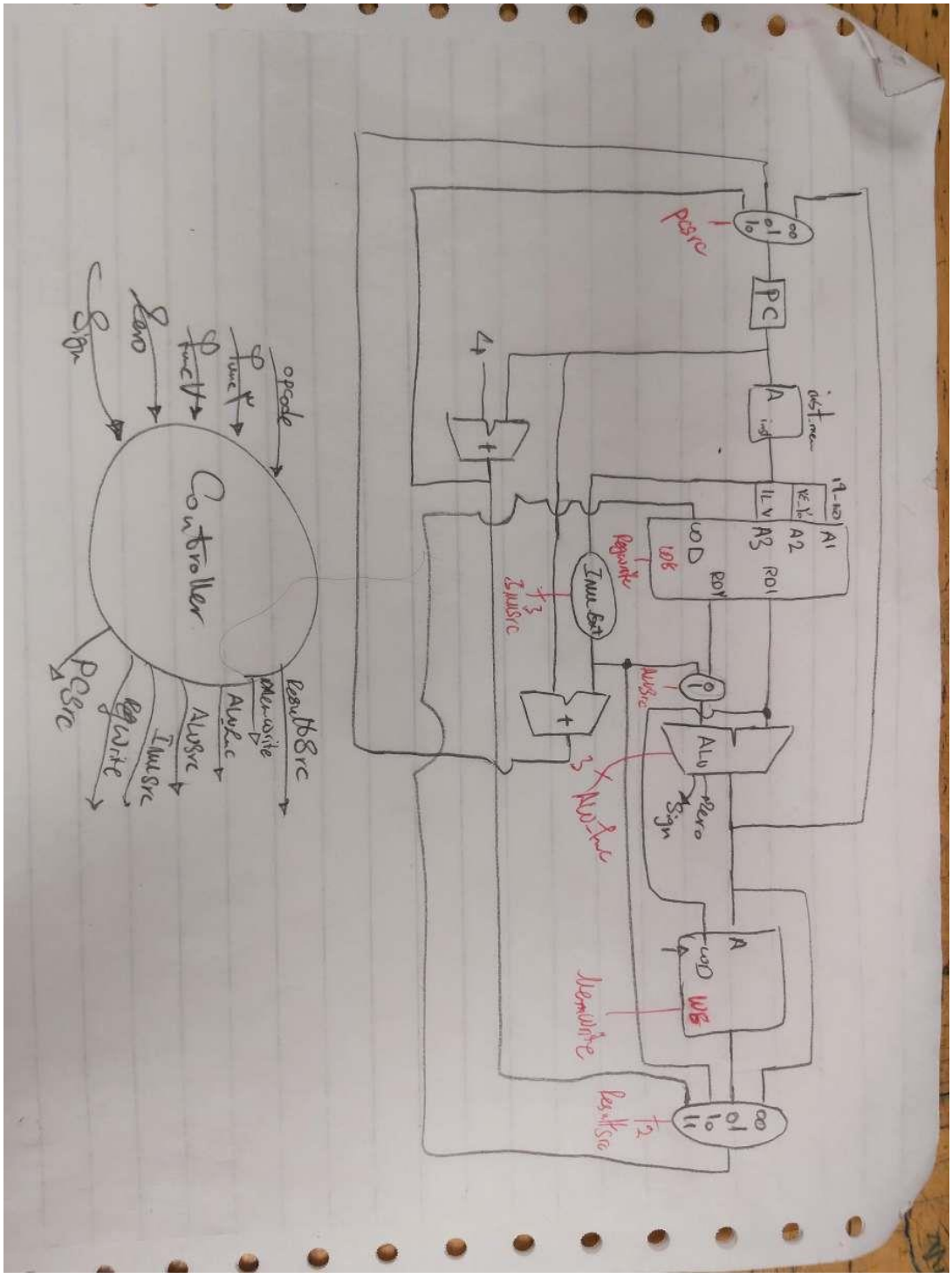
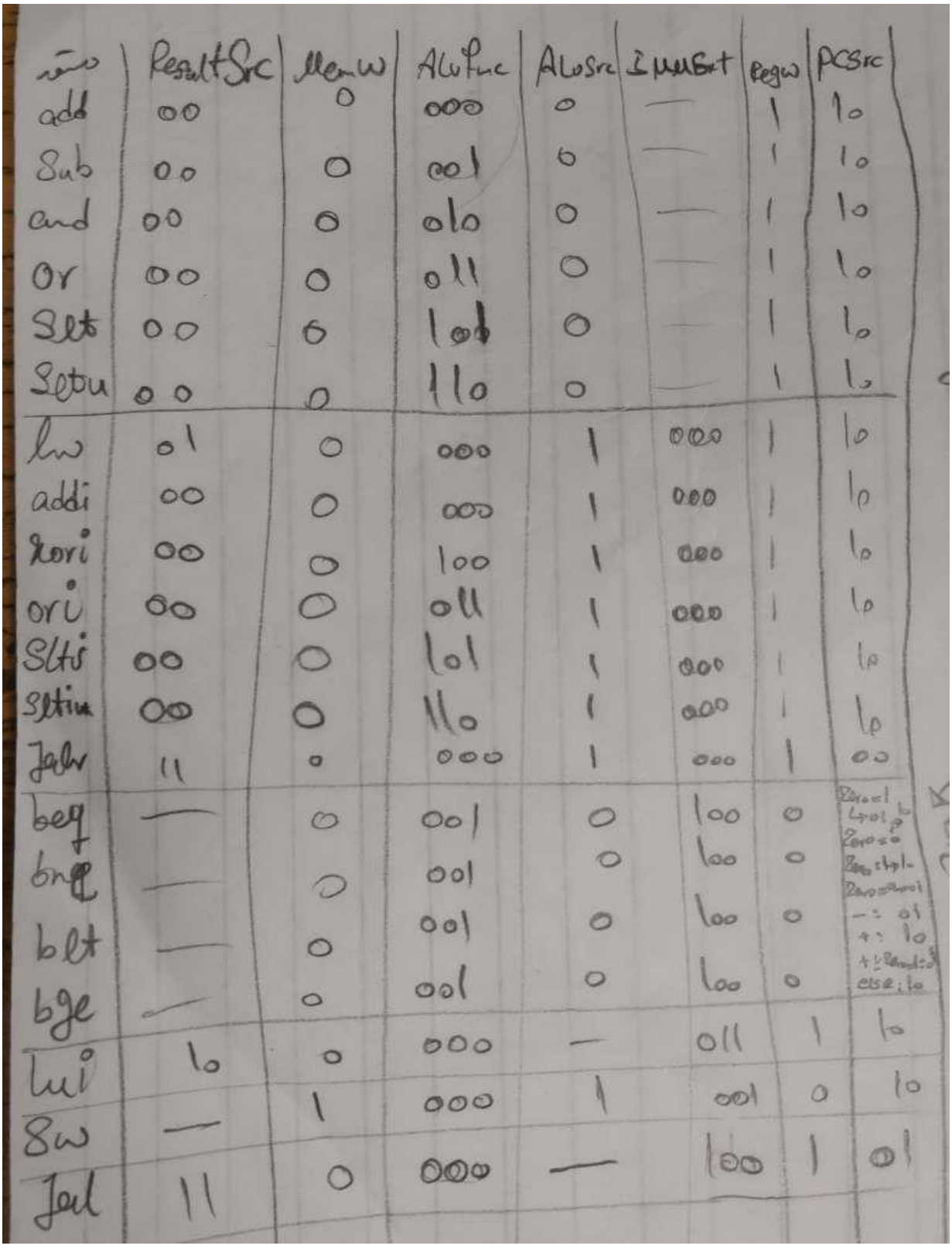
پروژه 2 معماری کامپیوتر

محمد طاها مجلسی 810101405

محمد سینا پرویزی مطلق 810101394

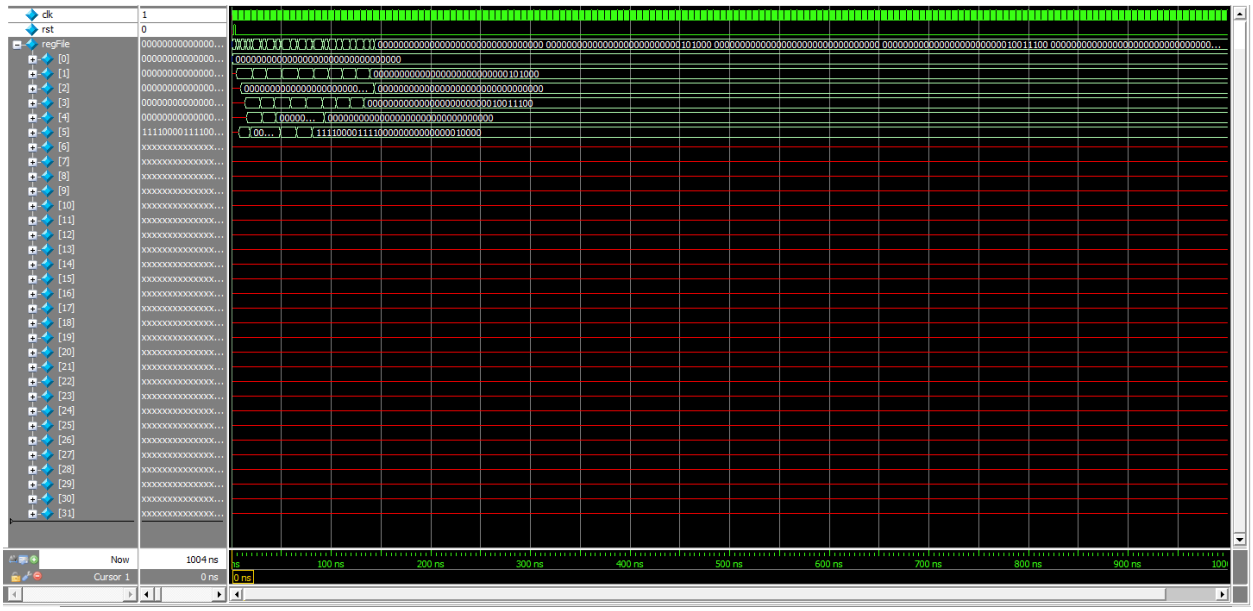


datapath

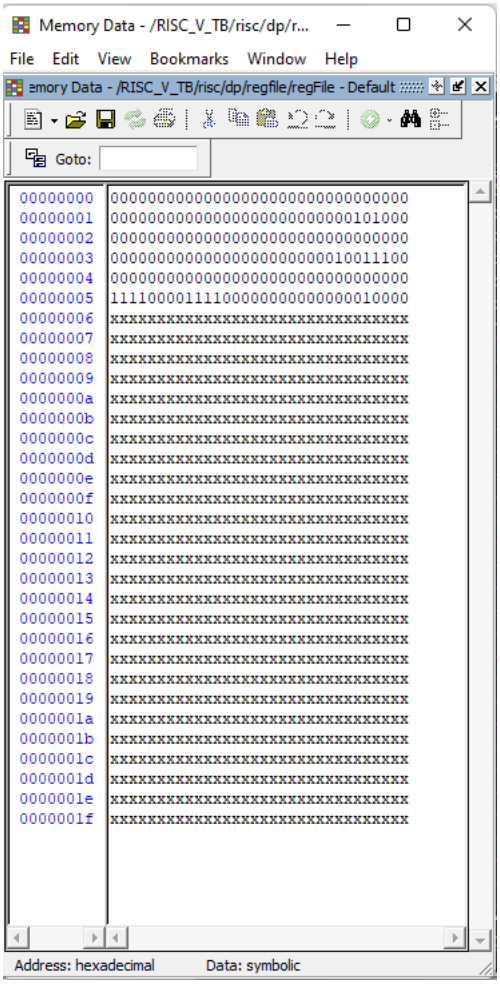


Instr	ResultSrc	MemW	ALUFunc	ALUSrc	ImmSrc	RegW	PCSrc
add	00	0	000	0	—	1	00
Sub	00	0	001	0	—	1	00
and	00	0	010	0	—	1	00
Or	00	0	011	0	—	1	00
Slt	00	0	101	0	—	1	00
Sltu	00	0	110	0	—	1	00
lw	01	0	000	1	000	1	00
addi	00	0	000	1	000	1	00
xori	00	0	100	1	000	1	00
ori	00	0	011	1	000	1	00
Slti	00	0	101	1	000	1	00
sltiu	00	0	110	1	000	1	00
Jalr	11	0	000	1	000	1	00
beq	—	0	001	0	100	0	Zero=1
bne	—	0	001	0	100	0	Zero=0
blt	—	0	001	0	100	0	-: 01
bge	—	0	001	0	100	0	+: 10
lui	10	0	000	—	011	1	00
Sw	—	1	000	1	001	0	00
Jal	11	0	000	—	100	1	01

Controller

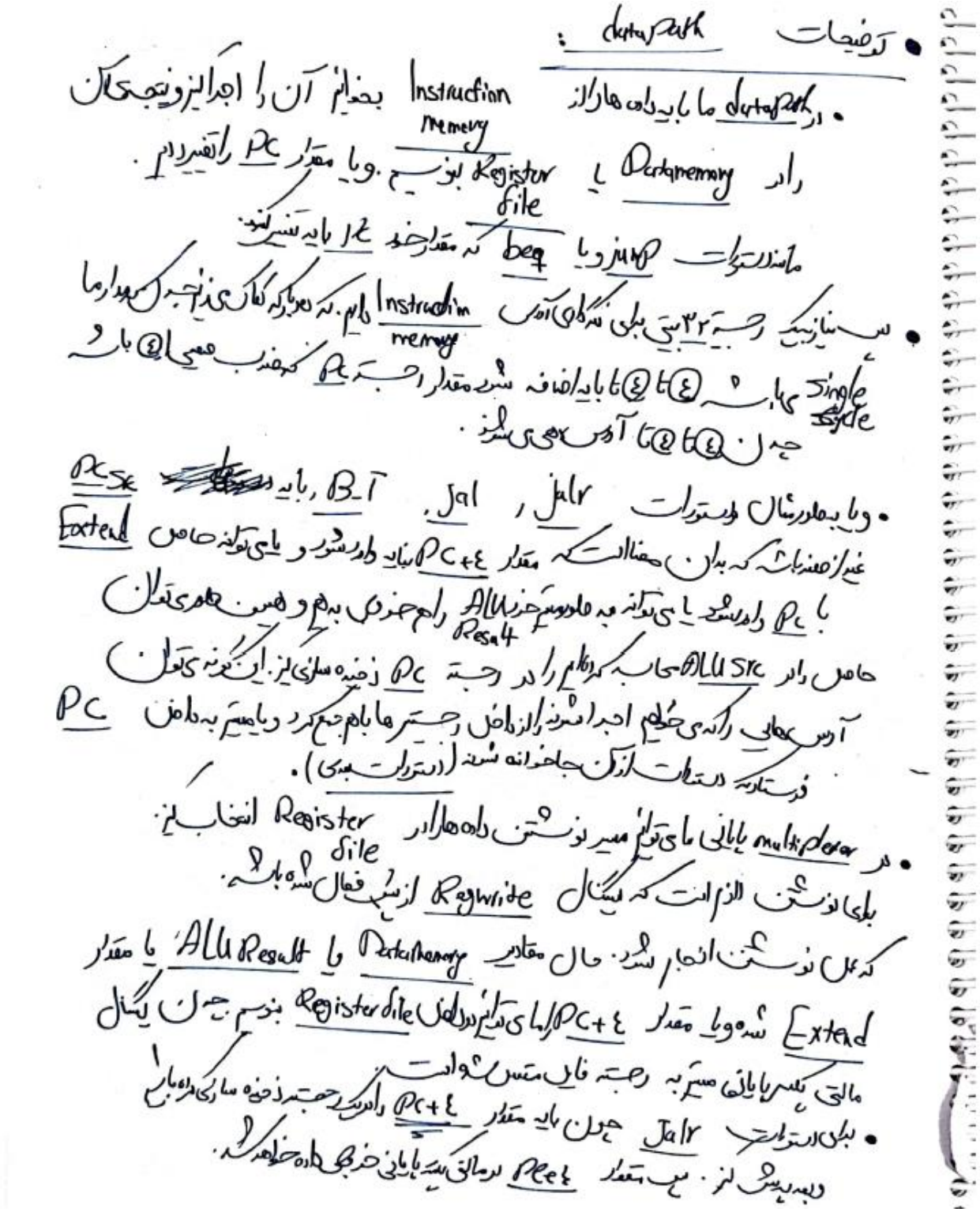


final result



Register file

مطاالب از اینجا به بعد را قبل از زدن پروژه یک چیز هایی نوشته بودیم که حالا میذاریم ولی خب تغییر یافته قطعا.



[redacted]

• ابتدا باید دستورات Jalr و Jal را اضافه کنیم که پیاده سازی کنیم :

مطابق با manual دستورات RISCV دستورات J type به این شکل هستند:

j offset → jal x0, offset Jump

(Jal) offset → jal x1, offset → Jump and link

• Jal — Jump and link یعنی به دستوری برو که Imm آن یا همان offset آن را قدر کرد و شیفت بده به PC اضافه کن ولی قبل از آن مقدار آدرس دستور بعدی را در رجیستر x1 باید ذخیره سازی کند.

• این دستورات برای صدا زدن توابع به کار برده خواهند شد.

• پس باید این مقدار PC target در رجیستر PC ذخیره خواهد شد (مثلاً برای return address جایی که باید برگردیم)

حال برای دستورات B_type مانند beq, bne, blt, bge باید ۲ تا رجیستر را دریافت کنیم مقادیر آن ها را باهم مقایسه کنیم و نتیجه ی آن ها را بر اساس آن ها مقدار PCSrc را اعمال کنیم. که یعنی یا مقدار PC+4 طبق معمول وارد رجیستر PC می شود و یا مقدار Imm+PC وارد رجیستر خواهد شد PC target.

در این حالت چیزی در رجیسترها نوشته نمی شود پس مقدار ResultSrc dont care خواهد بود. تنها چیزی که لازمه جواب است. چون دستور b_type طبق باید مقدار ImmSrc باید مقدار 010 بگیرد و سیگنال branch هم باید فعال باشد. از میان تمام دستورات فقط همین دستور (نوع دستور) سیگنال branch آن باید فعال باشد. و مقدار PCSrc هم باید در بعضی حالات تغییر کند یعنی بعضی وقت ها باید پرش کند بر اساس نوع دستور داده می شود برقرار باشد یا نه.

• برای دستورات U-type مانند دستور lui که باید مقدار (20) بیت را در یک رجیستر rd که تعیین می‌شود بنویسیم. در datapath باید کل ۳۲ بیتی را پس از Extend به حالتی که در خروجی پایانی کار بریم که زمانی که بخواهیم این دستور را اجرا کنیم مقدار ResultSrc باید مقدار (11) باشد که دیتا مستقیم به WD متصل باشد و همچنین سیگنال RegWrite هم باید فعال باشد. علاوه بر این سیگنال ImmSrc که بیانگر نوع برداشتن از دیتای Imm می‌باشد باید در حالت 100 باشد تا بتوان آن دیتا را در رجیستر مقصد (rd) ذخیره کرد.
PCSrc هم مطابق معمول باید 0 باشد که یعنی PC+4 را به کار می‌بریم.

برای دستورات SW یا Store word باید دیتایی که وارد می‌شود را سازی رجیستر مقصد را جمع مقدار و در پایان offset آن باشد.
سیگنال PCSrc باید 0 باشد مطابق معمول. ~~و ResultSrc هم باید 01 باشد که دیتایی که از روی Data memory است را بخوانیم و در رجیستر مقصد بنویسیم.~~

• و ResultSrc اهمیتی ندارد چون چیزی در رجیستر نوشته نخواهد شد.
و در پایان سیگنال ImmSrc باید 001 باشد که برای دستورات Store type می‌باشد و همچنین سیگنال memWrite که باید با آن در memory چیزی را بنویسیم هم باید 1 باشد چون به دنبال ذخیره‌سازی دیتا در آدرسی در memory هستیم.
یعنی مقدار رجیستر مقصد (rs1) به همراه دیتای extend شده با هم جمع خواهد شد و آدرس دیتا memory را می‌سازند و آدرس (rd) که همان دیتای 7-11 است باشد هم در آدرس نوشتن Register file قرار نخواهد گرفت که بتوان آن را نوشت و آدرس‌ها صحیح باشد.

- در دستورات I-type یا همان Immediate type باید دیتای را از رجیستر ها و به طور مستقیم از دستور بگیریم. و نتیجه‌ی آن را در یک رجیستر دیگر ذخیره سازی کنیم.
همان طور که می‌دانیم پس باید rs1 را بگیریم و به همراه آن دیتای Imm را هم از دستور بگیریم.
و نتیجه‌ی آن را که ALU به ما می‌دهد در رجیستر مقصد ذخیره کنیم. پس ResultSrc باید مقدار 00 را داشته باشد.
و ALUSrc هم باید مقدار 1 را داشته باشد. و همین طور سیگنال RegWrite باید فعال باشد.
و سیگنال Imm

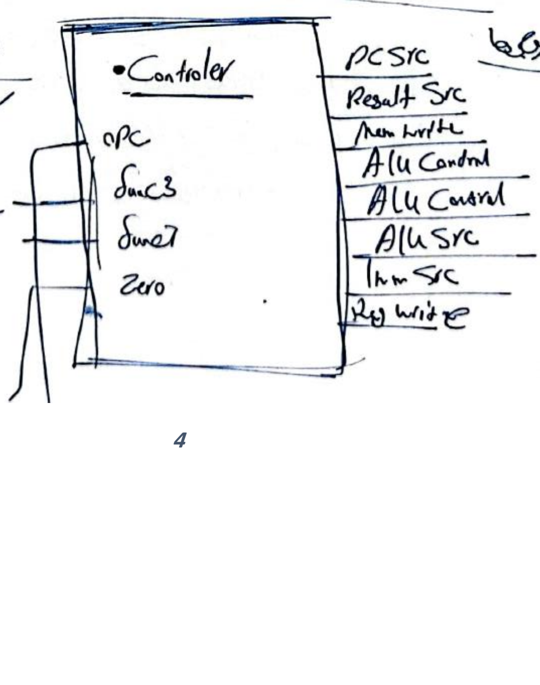
- در دستور lw باید مقداری را که در حافظه نوشته شده را در Reg file ذخیره سازی کند.
پس باید سیگنال ResultSrc ← 01 باشد که اطلاعات output حافظه را خروجی بدهد و ذخیره سازی در رجیستر مقصد انجام بگیرد. و همین طور مانند دستور قبلی آدرس آن از Imm + rs1 بدست خواهد آمد. یعنی این گونه محاسبه می‌شود. پس ALUSrc باید 1 باشد و سیگنال RegWrite هم باید فعال باشد که در زمان کلاک بعد از آن در رجیستر نوشته شود.

- و در پایان کنترل سیگنال‌هایی که دریافت می‌شوند به این شکل است:

ورودی‌ها

که البته یکی از دستورات به این funct3 و funct7 نیاز ندارد.

- خروجی‌ها



• پیاده سازی کنترلر 8

• همان طور که در کلاس مطرح شد این پردازنده ۳ تا کنترلر اصلی دارد:

که در پایان این ۳ تا باید با هم ترکیب شوند و کنترلر اصلی ساخته بشود:

1. Main Controler
2. Branch Controler
3. ALU Controler

• این ۳ تا کنترلر داریم.

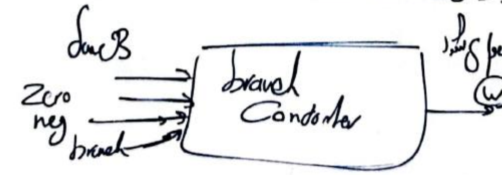
• ابتدا main Controler را بررسی می کنیم،

• ابتدا Branch Controler را بررسی می کنیم: • در این کنترلر باید بر اساس مقدار branch و مقادیر سیگنال های که در ادامه از ALU بر اساس مقایسه بیرون آمده بودند (سیگنال ها) باید مقدار PCSrc را مشخص کنیم که branch انجام شود یا نشود.

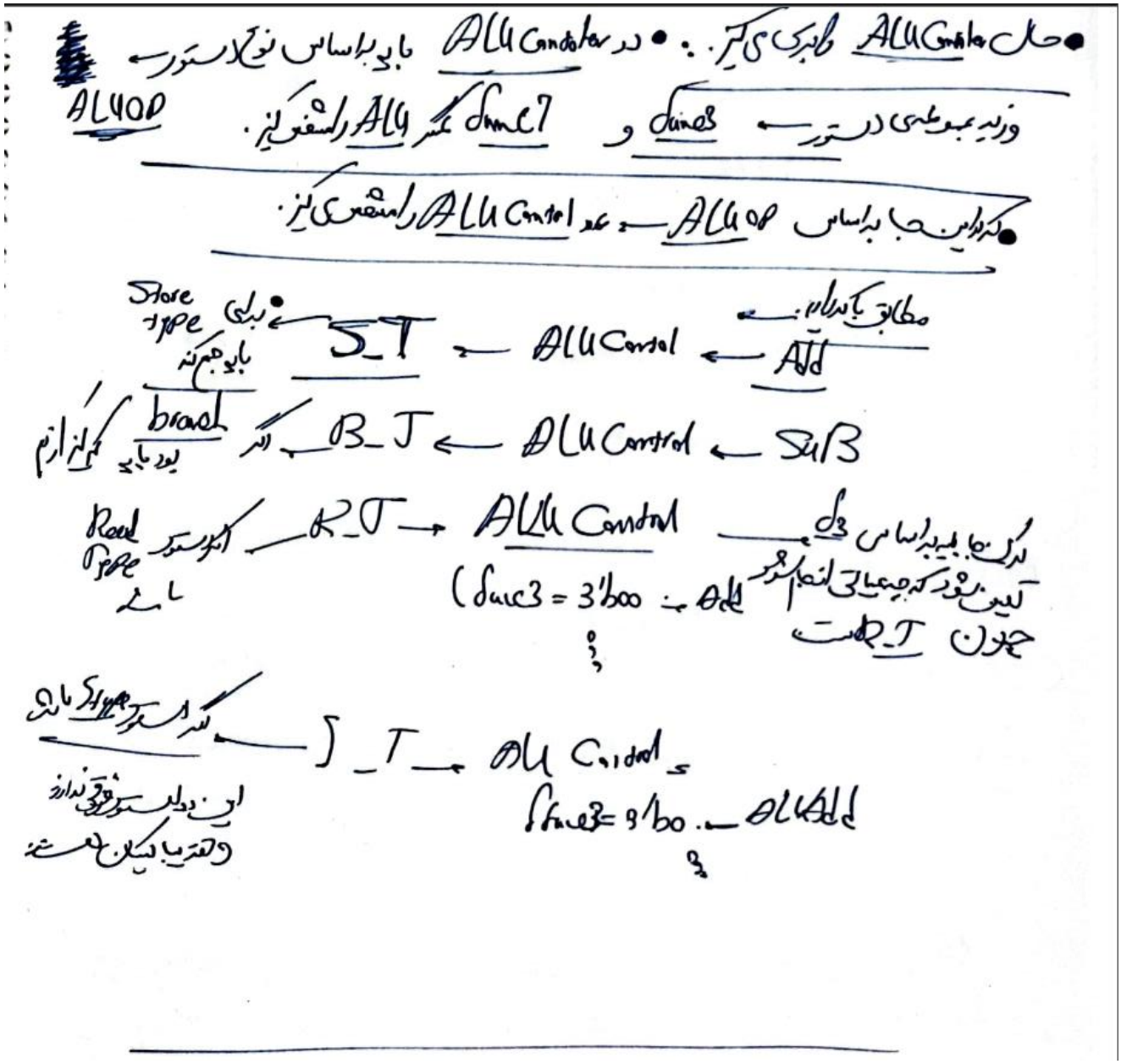
مطابق با کد داریم

Beq	← $w \Leftarrow branch \,\&\, zero$ →	اگر دو رجیستر مساوی بود
Bne	← $w \Leftarrow branch \,\&\, \sim zero$ ←	اگر دو رجیستر نامساوی بود
blt	← $w \Leftarrow branch \,\&\, neg$ ←	اگر کوچکتر بود
bge	← $w \Leftarrow branch \,\&\, (zero \mid \sim neg)$ ←	اگر بزرگتر یا مساوی بود

مقادیر ورودی به این کنترلر بدین صورت است



• می دانیم که بر اساس func3 معلوم می شود کدام دستور BType را داریم استفاده می کنیم.



• حال ALU Control را طراحی می‌کنیم. • در ALU Control باید براساس نوع دستور ← ALUOp

و در نتیجه بسته به نوع دستور ← func3 و func7 عملیات ALU را مشخص کنیم.

• که در این جا براساس ALUOp ← عملیات ALU Control را مشخص کنیم.

مطابق با جدول ←

Add → ALU Control → S_T ← • یعنی Store type باید جمع کند

SuB → ALU Control → B_T ← اگر branch بود باید کم (تفریق) کنیم

f3 ← ALU Control → R_T ← اگر دستور Read type بود

دیگر جا باید براساس f3 تعیین شود که چه عملیاتی انجام شود چون R_T است

(func3 = 3'b00 → Add

⋮

ALU Control → I_T ← اگر دستور I type باشد

func3 = 3'b0.. → ALU Add

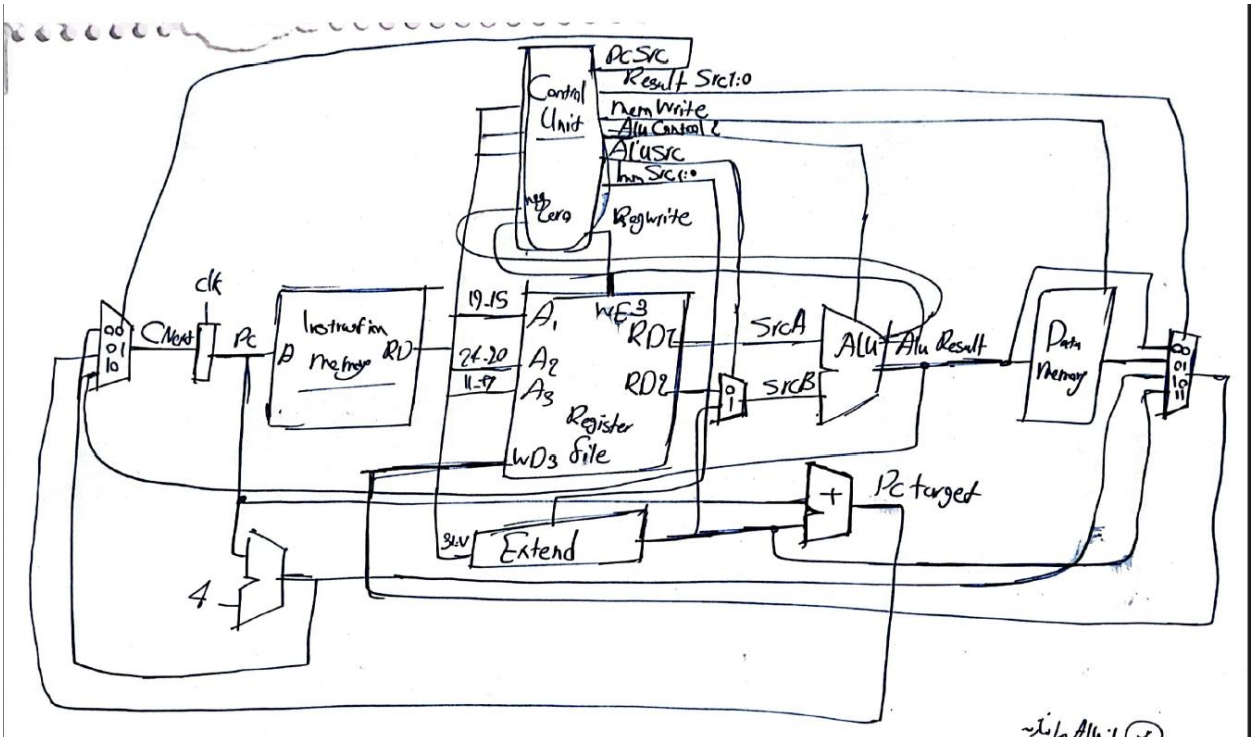
⋮

این دلیل ستون فوق ندارد و تقریبا یکسان هستند

• بخش main Controller : دیکد بخش جهش با بررسی opc و تعیین کردن نوع دستورات و سیگنال‌های کنترلی لازم آن‌ها

• جدول آنچه که می‌خواهد:

	PC Src	Result Src	mem write	Alu Control	Alu Src	Imm Src	Right Register	Branch	Jal	ALUop
RT	0	00	0		0	XX	0		0	10
I_T	0	00	0		1	000	0		0	11
Lw	0	01	0		1	000	0		0	00
Sw	0	X	1	000 (+)	1	001	0		0	00
Jal	1	10	0		X	011	0		1	XX
B_T	تعیین نشده	10	0	000 (-)	0	010	1		0	01
U_T	0	11	0		X	100	0		0	XX



* از ALU ما نیز سیگنال neg میگیریم که بتوانیم خاصل مقایسه‌کردن دو عدد را بدانیم