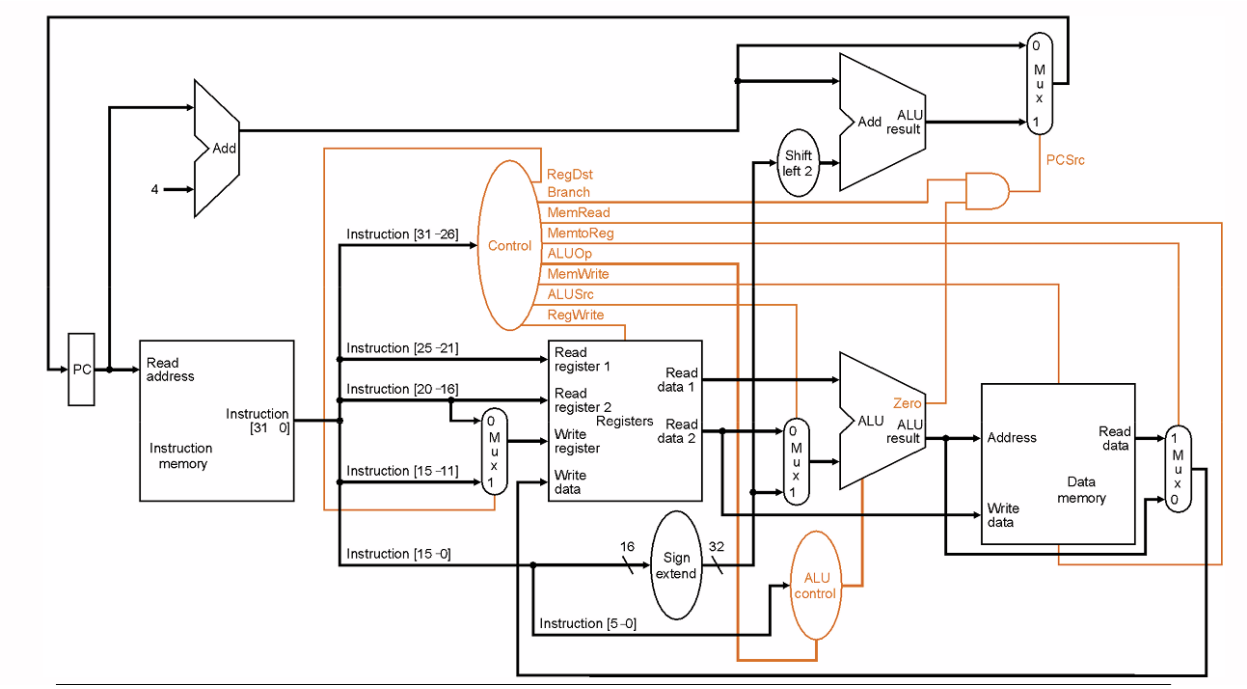
درس معماری کامپیوتر

تمرین سری چهارم ۱۴۰۴/۰۲/۱۸

متین باقری ۴۰۲۱۰۵۷۲۷

---

۱) با توجه به شکل زیر به این سوال پاسخ داده شده:



دستور swap $rs,$rt,imm:

قبل از write register در RF یک mux اضافه کرده که ورودی صفر آن خروجی mux قبلی و ورودی یک آن instruction[25-21] یا همان rs است. یک mux دیگر هم قبل از write data در Data memory قرار داده که ورودی صفر آن read data 2 و ورودی یک آن read data 1 از RF هستند. یک mux دیگر هم قبل از ورودی بالایی

ALU اضافه می‌کنیم که ورودی صفر آن read data 1 و ورودی یک آن imm sign extended هستند. سیگنال کنترلی آن‌ها تنها در صورت swap بودن دستور یک می‌شود.

signal	value
RegDst	x
Branch	0
MemRead	1
MemToReg	1
ALUop	00 (add)
memWrite	1
ALUsrc	0
RegWrite	1
swap	1

دستور addnz $rs,$rt,imm:

بیت‌های ReadData 2 از RF یا همان rt را با هم OR می‌کنیم و حاصل را به عنوان سیگنال regWrite در نظر می‌گیریم. قبل از write register در RF یک mux اضافه کرده که ورودی صفر آن خروجی mux قبلی و ورودی یک آن instruction[25-21] یا همان rs است. سیگنال سلکتور آن addnz بودن دستور است.

signal	value
RegDst	x
Branch	0
MemRead	0
MemToReg	0
ALUop	00(add)
memWrite	0
ALUsrc	1
RegWrite	OR(rt)
addnz	1

دستور loadpc $rd:

قبل از write data در RF یک mux اضافه کرده که ورودی صفر آن همان ورودی قبلی write data و ورودی یک آن pc بعد از جمع شدن با ۴ است. سیگنال سلکتور این mux را loadpc می‌نامیم.

signal	value
RegDst	1

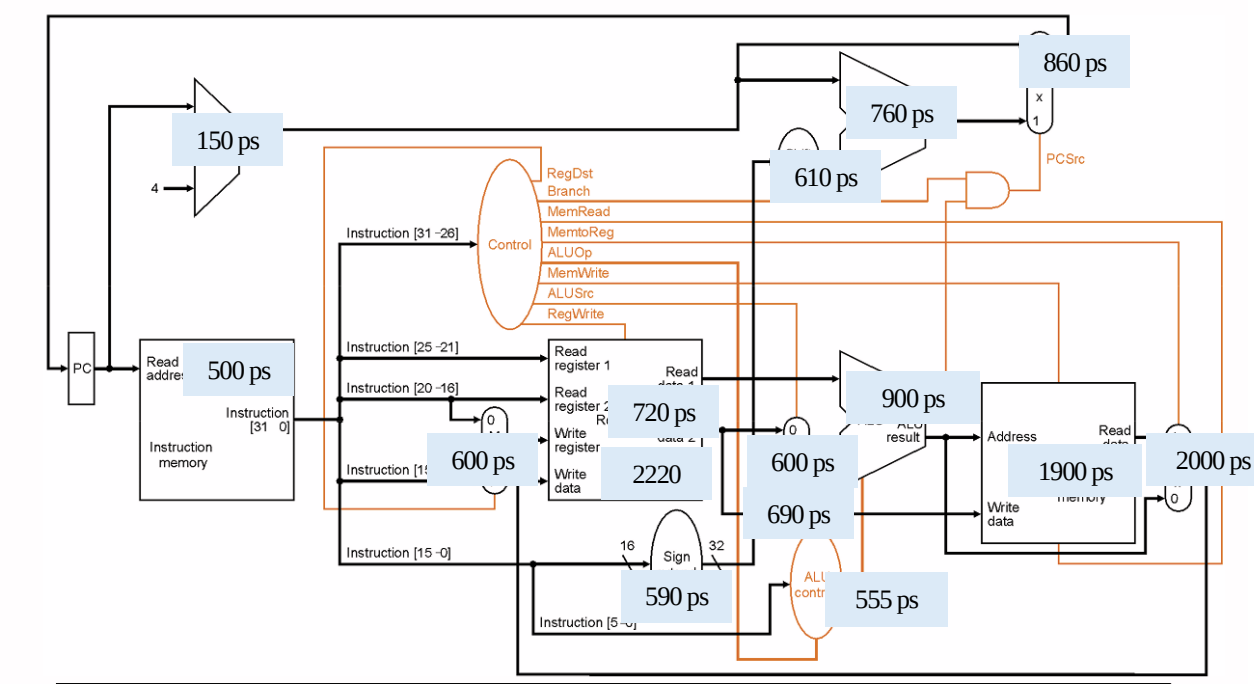
Branch	0
MemRead	0
MemToReg	x
ALUop	x
memWrite	0
ALUsrc	x
RegWrite	1
loadpc	1

دستور brsumz rs, rt, offset:

نیاز به اضافه کردن سخت‌افزار خاصی نداریم. در صورت یک شدن سیگنال zero دستور به درستی اجرا می‌شود. سیگنال های کنترلی به این صورت هستند:

signal	value
RegDst	x
Branch	1
MemRead	0
MemToReg	x
ALUop	00 (add)
memWrite	0
ALUsrc	0
RegWrite	0

۲) آ) تاخیر آماده شدن هر بخش را روی شکل مشخص کردیم:



بنابراین طولانی ترین دستور 2220 ps زمان می‌برد و زمان چرخه ساعت 2220 ps = 2.22 ns است.

ب) opcode در زمان 500 ps به Control Unit داده می‌شود. پس واحد کنترلی برای تولید 720 ps memWrite زمان دارد.

$$2220 - (1000+500) = 720 \text{ ps}$$

ج) سیگنال jump باید 100 ps قبل از کلاک بعدی آماده باشد. پس واحد کنترلی برای تولید آن 1620 ps زمان دارد.

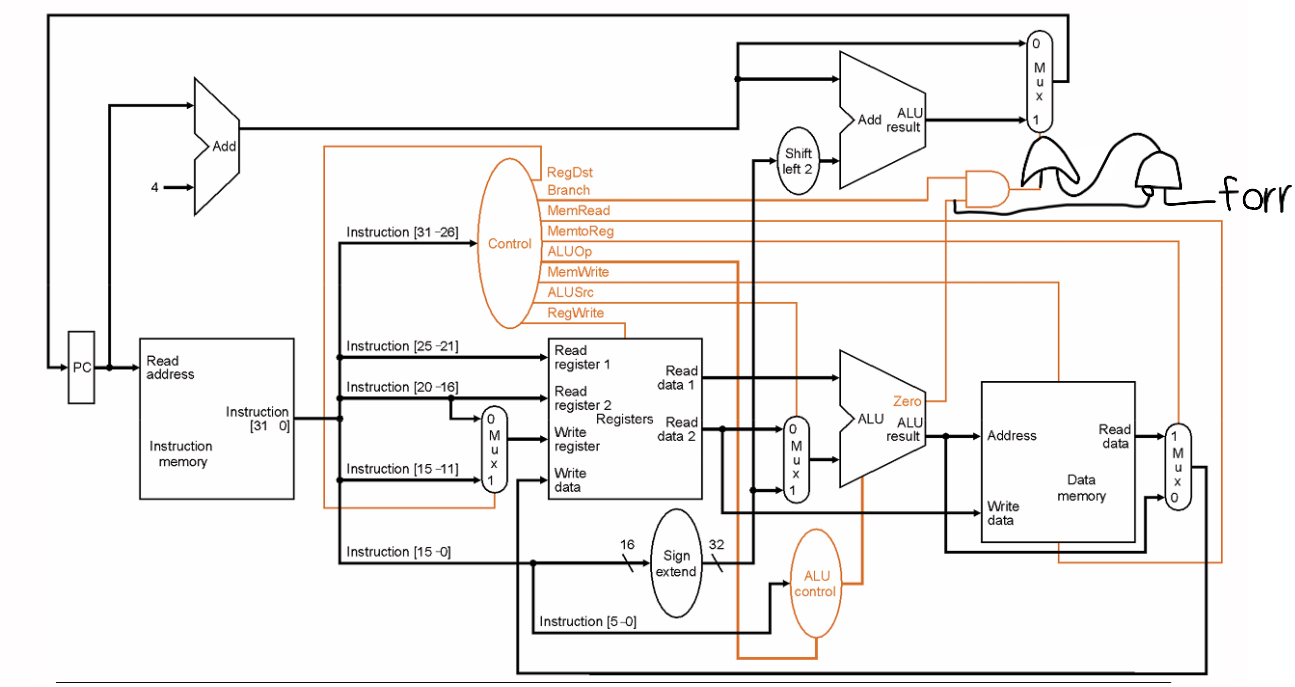
$$2220 - (500+100) = 1620 \text{ ps}$$

د) سیگنال ALUsrc باید در زمان 620 ps آماده باشد تا پس از گذشت 100 ps همزمان با آماده شدن خروجی RF و در زمان 720 نتیجه آن آماده باشد. پس واحد کنترلی برای آماده سازی آن 120 ps زمان دارد.

$$620 - 500 = 120 \text{ ps}$$

۳) آ) دستور از نوع I-Type است چون مقدار imm دارد.

ب) مجددا طبق این مدار داریم:



قبل از write register در RF یک mux اضافه کرده که ورودی صفر آن خروجی mux قبلی و ورودی یک آن instruction[25-21] یا همان rs است. بیت سلکتور آن سیگنال forr است که در صورتی که دستور forr باشد یک می‌شود. سپس گیت های and, or, not را مطابق آنچه در بالا سمت راست تصویر بالا رسم کردیم اضافه می‌کنیم تا بتوان به offset داده شده branch کرد.

signal	value
RegDst	x
Branch	0
MemRead	0
MemToReg	0

ALUop	01 (sub)
memWrite	0
ALUsrc	0
RegWrite	1
forr	1

۴) آ) برای دستور lui 16 بیت بالای رجیستر مقصد را مقدار imm داده شده و 16 بیت دیگر را 0 بگذارد. (البته اینکه 16 بیت imm در 16 بیت بالای رجیستر مقصد قرار بگیرد یا پایین، در شکل به طور دقیق مشخص نشده.)

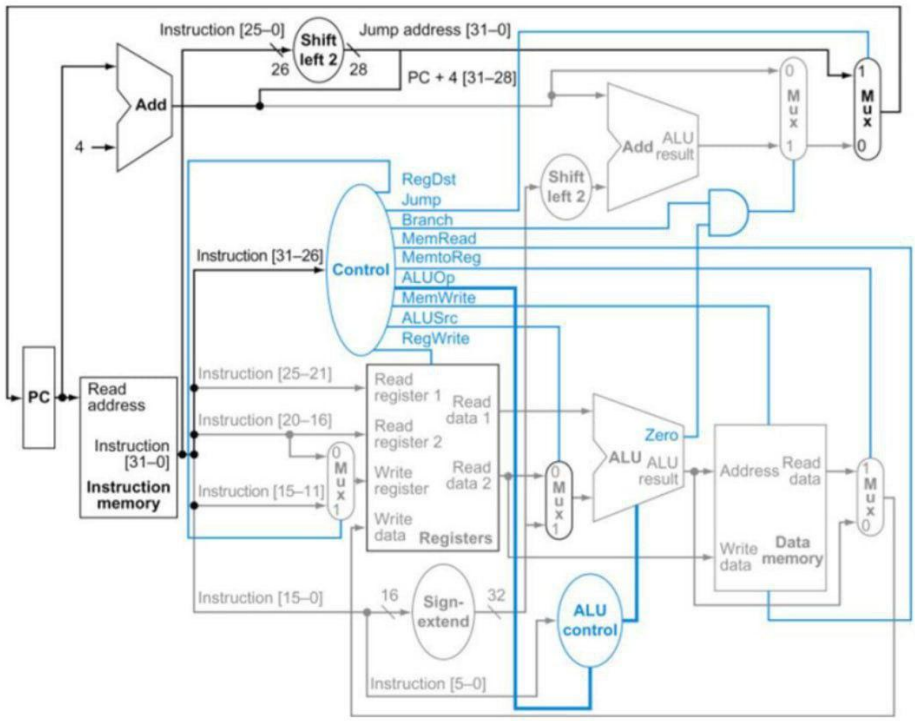
lui rs rt imm

| opcode (6 bits) | rs (5 bits) | rt (5 bits) | immediate (16 bits) |

ب)

Instr	RegDst	ALUSrc	Mem toReg	Reg Write	Mem Read	Mem Write	Branch	ALUOp ۱	ALUOp ۲	سیگنال جدید
R-type	۱	۰	۰	۱	۰	۰	۰	۱	۰	0
lw	۰	۱	۱	۱	۱	۰	۰	۰	۰	0
sw	x	۱	x	۰	۰	۱	۰	۰	۰	x
beq	x	۰	x	۰	۰	۰	۱	۰	۱	x
دستور جدید	0	x	x	1	0	0	0	x	x	1

۵) آ) در هر حالت دستوراتی که اجرای آن ها با مشکل مواجه می‌شود را نوشته ایم.



RegDest:

stuck at 0:

دستوراتی که باید در rd بنویسیم. R-Type

stuck at 1:

دستوراتی که باید در rt بنویسیم. I-type, lw

Jump:

stuck at 0:

دستورات J-type, jump, jal

stuck at 1:

همه دستورات غیر از دستورات J-type, jump, jal

Branch:

stuck at 0:

دستورات branch, beq, bne, …

stuck at 1:

همه دستورات غیر از دستورات jump و branch

MemRead:

stuck at 0:

همه دستوراتی که نیاز به خواندن از حافظه دارند. lw

stuck at 1:

دستوراتی که نیاز به نوشتن در حافظه دارند. sw

چون نمیتوان همزمان از حافظه خواند و در آن نوشت.

MemToReg:

stuck at 0:

همه دستوراتی که نیاز به خواندن از حافظه دارند. lw

stuck at 1:

همه دستوراتی که نیاز به نوشتن نتیجه ALU در RF دارند. R-type: add, sub, and, or, xor, slt, sll, srl, …

MemWrite:

stuck at 0:

دستوراتی که نیاز به نوشتن در حافظه دارند. sw

stuck at 1:

همه دستوراتی که نیاز به خواندن از حافظه دارند. lw

چون نمیتوان همزمان از حافظه خواند و در آن نوشت.

همچنین در همه دیگر دستورات (به جز sw) داده ناخواسته در مموری نوشته می‌شود و داده های مموری را خراب می‌کند.

ALUsrc:

stuck at 0:

دستوراتی که ALU نیاز به imm دارد. addi, andi, ori, xori, slti

stuck at 1:

دستوراتی که ALU نیاز به rt دارد. … ,R-type: add, sub, and, or, xor, slt, sll, srl

beq, bne, …

RegWrite:

stuck at 0:

همه دستوراتی که نیاز به نوشتن در RF دارند. … ,R-type: add, sub, and, or, xor, slt, sll, srl

addi, andi, ori, xori, slti, …

lw

stuck at 1:

همه دستورات غیر از دستورات بالا