فرض کنید برای هر عملیات جمع ۱۵ نانوثانیه، هر عملیات مکمل‌گیری ۱۰ نانوثانیه و هر عملیات شیفت ۵ نانوثانیه زمان مورد نیاز باشد. بیشترین تسریعی که عملیات ضرب Booth نسبت به ضرب Add & Shift برای اعداد ۸ بیتی دارد چقدر است؟

تقسیم زیر را با روش Restoring انجام دهید.

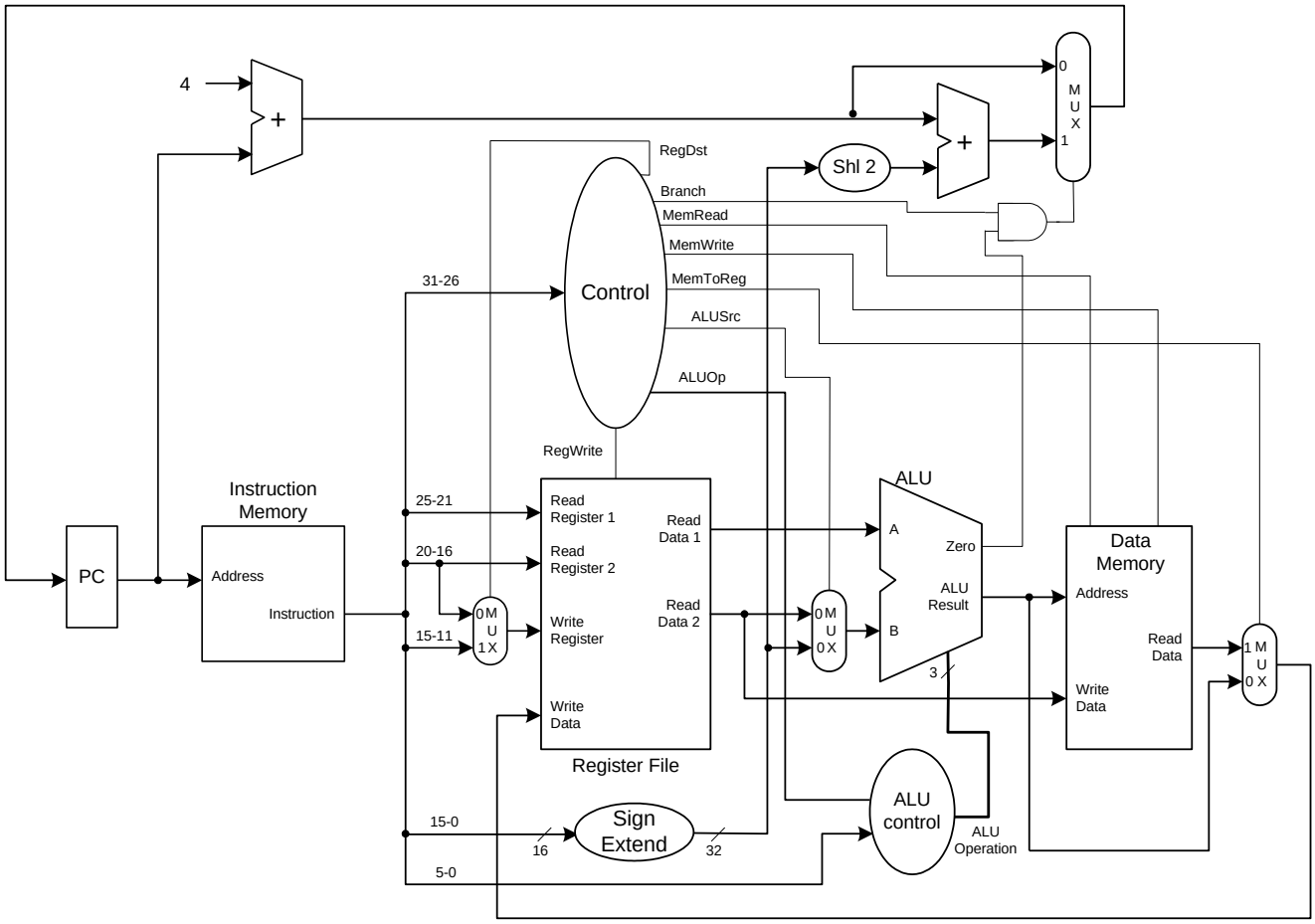
$X = (011011)_2 = 27, D = (0110)_2 = 6$

نمایش ممیز شناور عدد $-11\frac{2}{15}$ به صورت IEEE-754 Single Precision را به دست آورید.

جدول زیر نرخ حضور دستورات و متوسط تعداد سیکل مصرفی هر یک از دستورات را در یک برنامه‌ی Benchmark مشخص نشان می‌دهد. فرض کنید دستورات تقسیم ممیزشناور که نتیجه‌ی خارج قسمت آن از ۱ کمتر است را می‌توان با ۲ دستور ضرب ممیزشناور و یک دستور جمع ممیز شناور جایگزین نمود. برای این منظور باید یک دستور تفریق ممیزشناور و یک دستور پرش شرطی را به کار برد که مشخص شود آیا می‌توان از این جایگزینی استفاده نمود یا خیر. اگر برای یک برنامه‌ی Benchmark مقدار CPI برابر 3.5 شود، چند درصد از دستورات تقسیم ممیزشناور برنامه جایگزین شده‌اند؟

Instruction Type	Fixed-Point ALU ops.	Load	Store	Branch	Floating-Point Add/Sub	FP Mult	FP Div
Frequency	15%	19%	10%	18%	22%	10%	6%
Clock cycle count	1	2	2	2	3	5	24

شکل زیر مسیر داده و کنترلر پردازنده‌ی MIPS را در حالت تک مرحله‌ای نشان می‌دهد. حداقل تغییرات لازم را در مسیر داده و کنترلر اعمال کنید تا پردازنده توانایی اجرای دستورات wai $i (آدرس دستور جاری را در $i قرار می‌دهد) و *jr $i* (این دستور به آدرسی از حافظه که درون $i قرار دارد پرش می‌کند) را داشته باشد. توضیح دهید که آیا می‌توان با این دو دستور فراخوانی تابع و بازگشت تابع را پیاده‌سازی کرد یا خیر؟



دو عدد A و B به صورت ممیزشناور با دقت ساده در استاندارد IEEE نمایش داده شده اند. حاصل A + B را به صورت ممیزشناور و در همان استاندارد محاسبه کرده نمایش دهید. مقدار عددی معادل حاصل را نیز به دست آورید.

A = DCAB4358 B = 4432BB56

قانون آمدال (Amdahl's Law) برای نمایش تسریع به‌دست آمده در یک سیستم به صورت زیر ارائه شده است.

$$Speedup = \frac{Execution\ time\ before\ improvement}{Execution\ time\ after\ improvement}$$

$$Execution\ time\ after\ improvement = \frac{Execution\ time\ affected\ by\ improvement}{Amount\ of\ improvement} + Execution\ time\ unaffected$$

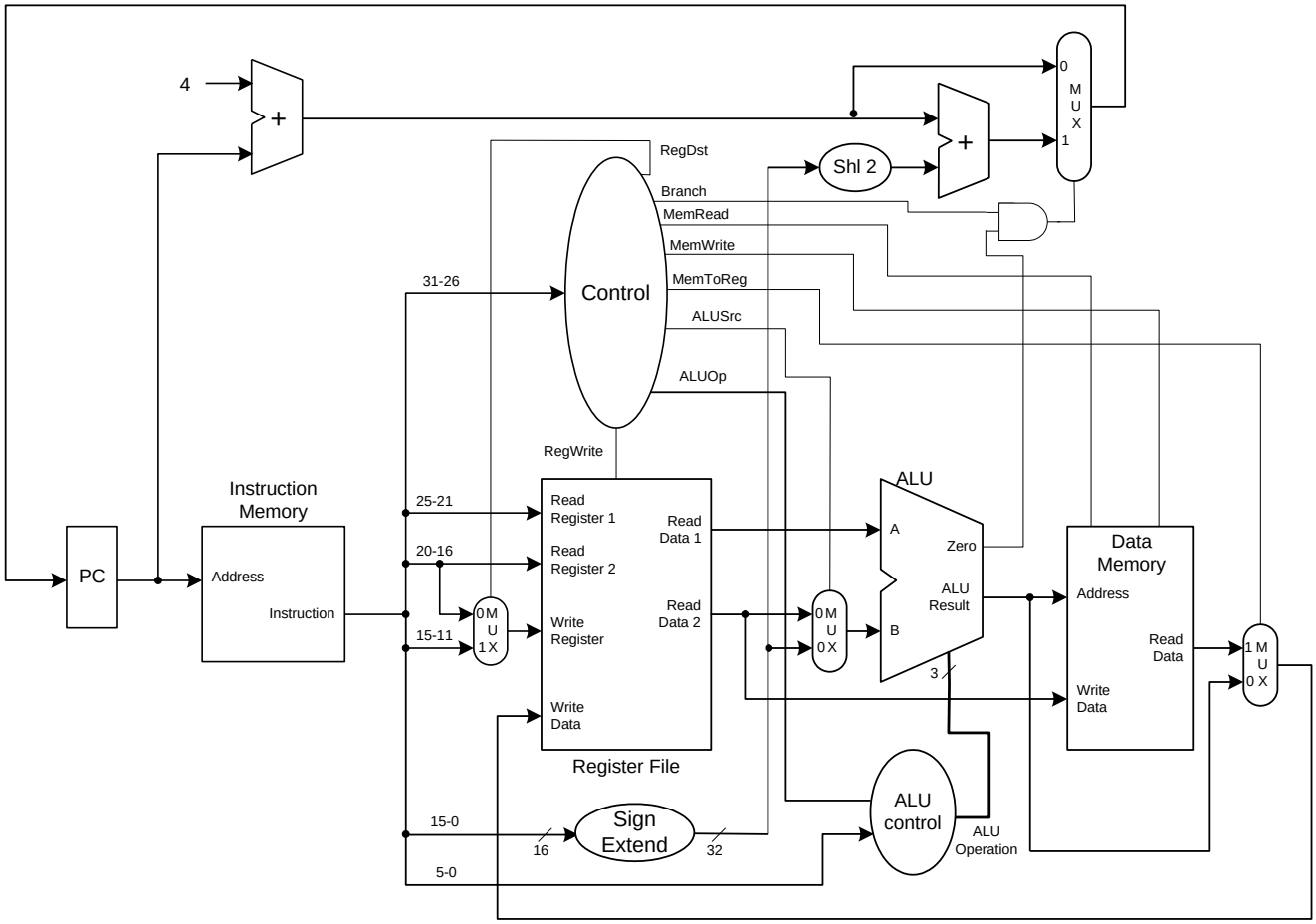
فرض کنید برای بهبود یک ماشین با دو انتخاب مواجه هستیم: اجرای ۴ بار سریع‌تر دستور ضرب و اجرای ۲ بار سریع‌تر دستور خواندن از حافظه. فرض کنید یک برنامه با زمان اجرای ۱۰۰ ثانیه به صورت متوالی روی این ماشین اجرا می‌شود. از این زمان 20% برای دستورات ضرب، 50% برای دستورات خواندن حافظه و بقیه برای سایر دستورات مصرف می‌شود. میزان تسریع به دست آمده را در سه حالت زیر به‌دست آورید.

الف– فقط دستورات ضرب بهبود پیدا کند.

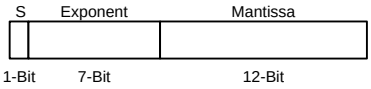
ب– فقط دستورات خواندن از حافظه بهبود پیدا کند.

ج– دستورات ضرب و خواندن از حافظه هر دو بهبود پیدا کند.

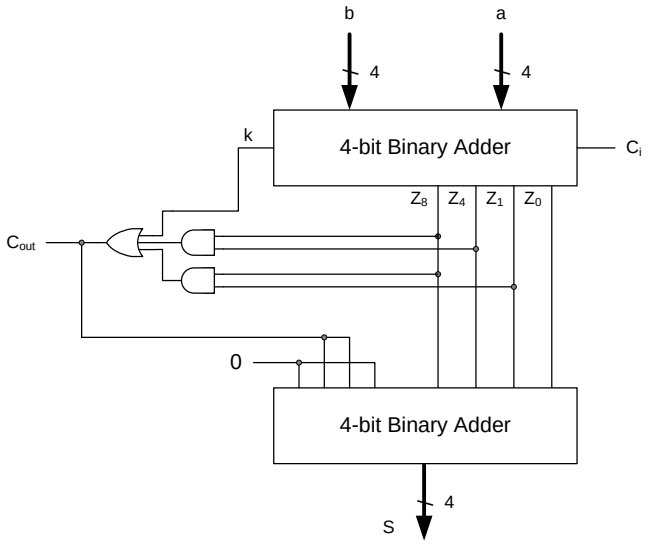
شکل زیر مسیر داده و کنترلر پردازنده‌ی MIPS را در حالت تک مرحله‌ای نشان می‌دهد. حداقل تغییرات لازم را در مسیر داده و کنترلر اعمال کنید تا پردازنده توانایی اجرای دستورات *push $i* و *compl $i, $j* را داشته باشد. دستور *compl* مکمل ۱ رجیستر $j را درون $i قرار می‌دهد.



نمایش ممیز شناور زیر را در نظر بگیرید. بزرگ‌ترین و کوچک‌ترین عدد مثبت قابل نمایش در این سیستم را مشخص کنید. فرض کنید که از روش Implicit One برای نمایش مانتیس و روش Biased Exponent-64 (جمع توان‌ها با ۶۴) برای نمایش توان استفاده شده است.



شکل زیر یک جمع‌کننده‌ی یک رقمی BCD را نشان می‌دهد. اگر تاخیر گیت‌های پایه (and, or) برابر ۱ باشد و از تاخیر گیت‌های not صرف‌نظر کنیم، تاخیر یک جمع‌کننده‌ی BCD ۴ رقمی چقدر است؟
فرض کنید جمع‌کننده‌های ۴ بیتی به صورت Carry Propagation Adder ساخته شده‌اند.



پردازنده‌ای با مجموعه‌ی دستورات زیر را در نظر بگیرید.

Operation	Frequency	Clock Cycle
ALU	40%	1
Load Word	25%	2
Store Word	15%	2
Branch	20%	2

اضافه کردن یک دستور با مود آدرس‌دهی register-memory به یک ماشین load-store ممکن است مفید باشد. برای مثال می‌توان دو دستور

```
lw      R1, 4(R3)
add     R2, R2, R1
```

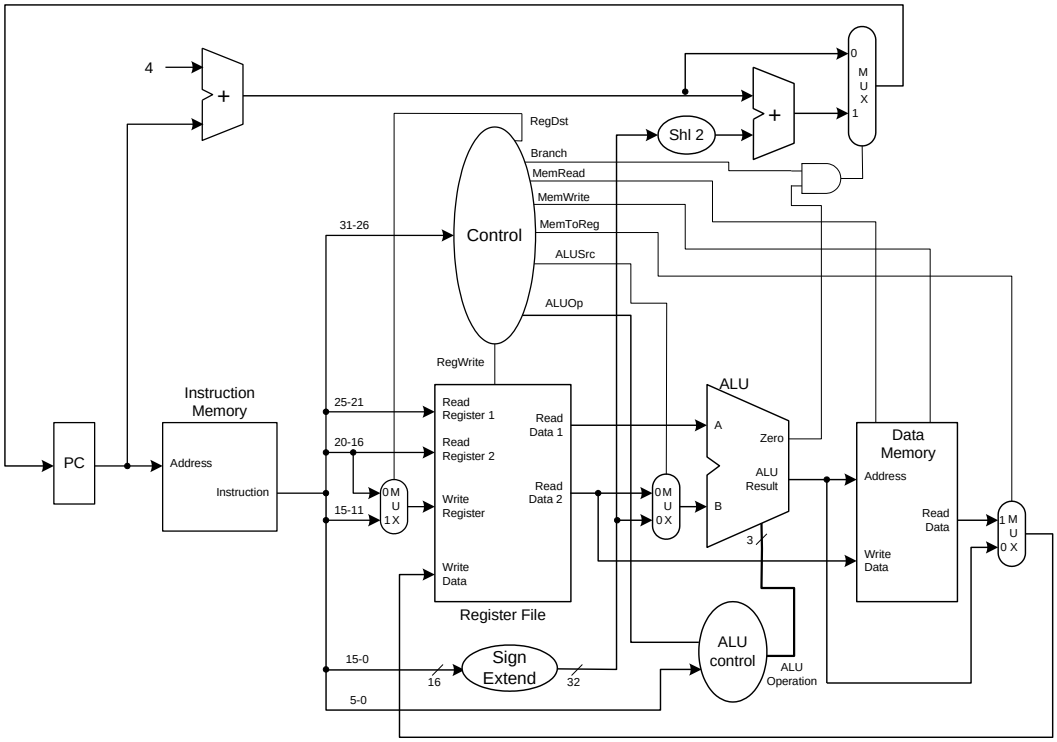
را با دستور addm R2, (R3) جایگزین نمود.

فرض کنید اضافه کردن این دستور –که اجرای آن ۳ سیکل ساعت طول می‌کشد- سبب شده است که تعداد سیکل‌های لازم برای اجرای دستور Branch از ۲ به ۳ افزایش یابد. اگر ۱۵ درصد از تعداد دستورات load-word را بتوان با این دستور جدید جایگزین نمود، آیا تغییر در ساختار پردازنده مفید هست یا خیر؟

شکل زیر مسیر داده و کنترلر پردازنده‌ی MIPS را در حالت تک مرحله‌ای نشان می‌دهد. حداقل تغییرات لازم را در مسیر داده و کنترلر اعمال کنید تا پردازنده توانایی اجرای دستورات skip-next $i, $j و csr adr را داشته باشد. دستور اول در صورت برابر بودن $i و $j از روی دستور بعدی پرش می‌کند و دستور دوم یک نوع خاص فراخوانی تابع است که آدرس برگشت تابع را در اولین آدرس تابع می‌نویسد.

یک ایراد این روش فراخوانی تابع را بنویسید. به نظر شما برای برگشت از این تابع به چه نوع دستوری نیاز داریم؟

(همیشه اولین آدرس تابع را برای ذخیره‌سازی آدرس برگشت خالی می‌گذاریم)


4
+
RegDst
Shl 2
+
0
M
U
X
1
Branch
MemRead
MemWrite
MemToReg
ALUSrc
ALUOp
31-26
Control
RegWrite
Instruction
Memory
PC
Address
Instruction
25-21
20-16
15-11
0
M
U
1
X
Read
Register 1
Read
Register 2
Write
Register
Write
Data
Read
Data 1
Read
Data 2
Register File
ALU
A
B
Zero
ALU
Result
3
0
M
U
0
X
Data
Memory
Address
Read
Data
Write
Data
1
M
U
0
X
15-0
16
Sign
Extend
32
5-0
ALU
control
ALU
Operation

سؤال 4 -

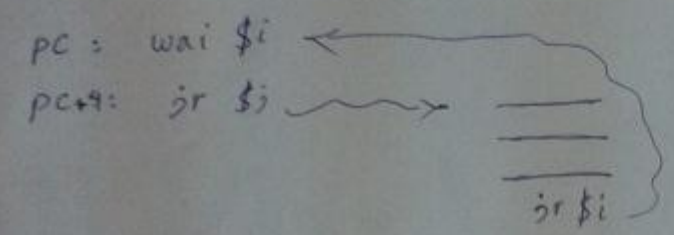
CPI = 2×5 + 1×3 + 1×3 + 1×2 = 18 cc برای دستورات تقسیم جدید

$$3.5 = 0.15\times1 + 0.19\times2 + 0.1\times2 + 0.18\times2 + 0.22\times3 + 0.1\times5 + (0.06-x)\times24 + x\times18$$

$$\Rightarrow \quad 3.69 - 6x = 3.5 \quad \Rightarrow \quad x = 0.032 \quad \Rightarrow \text{درصد کاهش} = \frac{0.032}{0.06} \simeq 53\%$$

سؤال 5 -

با استفاده از wai و jr نمی‌توان دستور فراخوانی و بازگشت تابع f پیاده‌سازی کرد. زیرا wai مقدار آدرس دستور جاری f ذخیره می‌کند و در صورت استفاده از این نوع فراخوانی برنامه در حلقه‌ی بی‌نهایتی می‌افتد:



سؤال 6 -

$$A = \underbrace{1}_{S_1}\ \underbrace{10111001}_{E_1}\ \underbrace{01010110\ 10000110\ 10110000}_{M_1}$$

$$B = \underbrace{1}_{S_2}\ \underbrace{00010000}_{E_2}\ \underbrace{11001010\ 11101101\ 01011110}_{M_2}$$

مراحل جمع/تفریق در FP:

1- Alignment: شیفت مانتیس عدد با توان کمتر به راست به اندازه‌ی اختلاف توان‌ها به منظور یکسان سازی توان دو عدد

2- جمع یا تفریق مانتیس:

3- post_normalize . نرمالیزه کردن مانتیس حاصل جمع/تفریق

جواب:

$E_1 - E_2 = 185 - 16 = 174$

پس باید مانتیس B به 174 بیت به راست شیفت بدهیم! (اعداد A و B در رنج هم نیستند به همین دلیل این مشکل پیش می‌آید) $\Leftarrow$

$A + B \simeq A$

(عدد بزرگ ← عدد کوچک)

سوال 7-

الف)

$$speedup_1 = \frac{100}{\frac{20}{4} + 80} = \frac{100}{85}$$

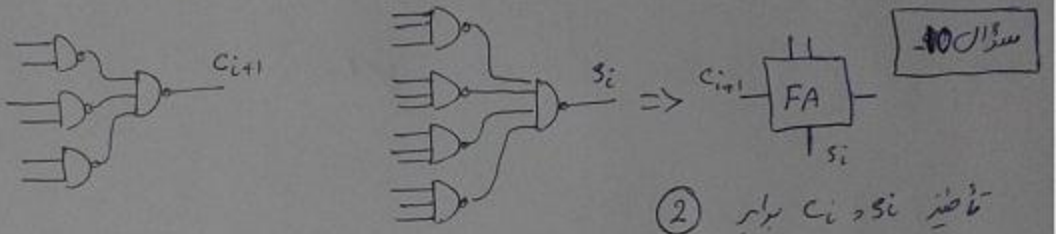
ب)

$$speedup_2 = \frac{100}{\frac{50}{2} + 50} = \frac{100}{75}$$

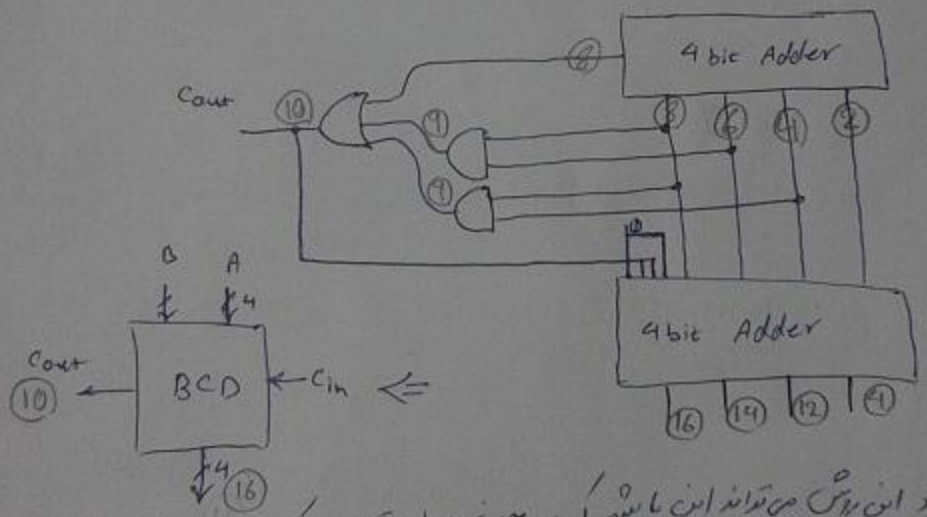
ج)

$$speedup_3 = \frac{100}{\frac{50}{2} + \frac{20}{4} + 30} = \frac{100}{60}$$

سوال 10-



تاخیر S_i و C_i برابر (2)



سوال 11-

یک ایراد این روش می‌تواند این باشد که در حین اجرای تابع ممکن است مقدار ذخیره شده در اولین آدرس تابع را تغییر دهیم و برنامه برای برگشت از تابع دچار مشکل شود.

برای برگشت می‌توان از دستوری مثل jmem adr استفاده کرد که PC را برابر محتوای ذخیره شده در حافظه‌ی با آدرس adr قرار می‌دهد.