بسمه تعالی

پاسخ تمرین یازدهم درس معماری کامپیوتر

نیم‌سال اول ۰۱-۰۰

۱. در مورد شیوه‌های آدرس‌دهی به سوالات زیر پاسخ دهید.

الف) شیوه‌های آدرس‌دهی‌ای که در آن‌ها، از ثبات‌ها استفاده می‌شود را نام برده و مثال بیاورید.

آدرس‌دهی‌های ثباتی مستقیم، ثباتی غیرمستقیم، خود افزایشی یا خود کاهشی، شاخص‌دار، نسبی با ثبات پایه و شاخص‌دار با ثبات پایه

اطلاعات بیشتر در لینک

ب) کدام‌یک از شیوه‌های آدرس‌دهی امکان استفاده از اشاره‌گرها (pointer) را می‌دهد؟

در شیوه‌های آدرس‌دهی ضمنی (Implicit)، بلافصل (Immediate)، حافظه‌ای مستقیم (Memory Direct) و ثباتی مستقیم، ما از اشاره‌گرها استفاده نمی‌کنیم.

پ) کدام‌یک از شیوه‌های آدرس‌دهی در افزایش سرعت برنامه‌ها تاثیر مستقیم دارند؟

هر کدام از شیوه‌های آدرس‌دهی شاخص‌دار، آدرس‌دهی با ثبات پایه و آدرس‌دهی شاخص‌دار با ثبات‌پایه، برای افزایش سرعت برنامه‌ها به مجموعه روش‌های آدرس‌دهی افزوده شده‌اند. بدون داشتن این شیوه‌ها هم ما می‌توانیم تمام قابلیت‌هایی را که لازم داریم برای برنامه‌ی خود پیاده‌سازی کنیم اما با داشتن این شیوه‌های آدرس‌دهی می‌توان دو یا چند دستور از برنامه را توسط یک دستور جایگزین انجام داد.۱۱

ت) توضیح دهید آیا امکان دارد در یک دستور سه آدرسی، هر عملوند سه شیوه مختلف آدرس‌دهی داشته باشند؟ با ذکر مثال توضیح دهید.

بله امکان دارد، در مثال زیر برای عملوند اول، از شیوه‌ی آدرس‌دهی شاخص‌دار با ثبات پایه استفاده شده، در عملوند دوم از شیوه‌ی آدرس‌دهی نسبی با ثبات پایه استفاده شده و در عملوند سوم آدرس شاخص‌دار مورد استفاده قرار گرفته‌است. در این دستور دو عملوند دوم و سوم را با هم جمع می‌کند و حاصل را در عملوند اول ذخیره می‌کند.

```
ADD [BP + DI], [BP]+2, [SI]
```

۲. سوالات زیر را همراه با دلیل پاسخ دهید:

الف) یک کامپیوتر دارای حافظه به ابعاد ۱۶ * ۱۶۳۸۴ است. اگر هر دستور در دو خانه متوالی حافظه قرار گرفته باشد، اندازه‌ی ثبات‌های AR, IR, AC, PC, DR را بیابید.

۱۶۳۸۴ = $۲^{۱۴}$ = تعداد خانه‌های کامپیوتر ← طول آدرس ۱۴ بیتی‌ست.

۱۶ = طول یک کلمه

ثبات‌هایی که در خود آدرس نگهداری می‌کنند مثل PC و AR، ۱۴ بیتی خواهند بود.

ثبات‌هایی که در خود داده ذخیره می‌کنند دارای طولی برابر با یک کلمه خواهند بود. پس طول ثبات‌های DR و AC، ۱۶ بیتی خواهند بود.

ثبات IR دستورات را در خود ذخیره می‌کند. از آنجا که در صورت سوال گفته شده طول دستور دو کلمه است، پس طول این ثبات ۳۲ بیتی خواهد بود. (از آنجا این ثبات در صورت سوال به اشتباه DR نوشته شده بود، اگر پاسخی داده نشده باشد مشکلی نیست و بارمی به آن اختصاص داده نشده. به جهت تکمیل توضیحات این بخش در پاسخ‌نامه اضافه شده است)

ب) هریک از اعمال زیر حداقل در چند کلاک انجام می‌شوند؟ دلیل خود را ذکر کنید.

$a)\ IR \leftarrow M[PC]$

دو کلاک:

AR ← PC
IR ← M[AR]

$b)\ AC \leftarrow AC + TR$

یک کلاک: در صورتی که تصور شود یک پایه‌ی ورودی ALU ثبات AC است و پایه‌ی دیگر ثبات TR، در اینصورت، در یک کلاک این محاسبه به پایان می‌رسد و خروجی در AC ذخیره خواهد شد.

AC ← AC + TR

سه کلاک: در صورتی که دو ثبات O1 و O2 ورودی‌های ALU باشند، به دو کلاک اضافه نیاز پیدا خواهیم کرد که قبل از عمل جمع، مقدار داخل ثبات‌های AC و TR را در ثبات‌های ورودی ALU لود کنیم.

O1 ← AC
O2 ← TR
AC ← O1 + O2

$c)\ DR \leftarrow DR + AC$

دو کلاک: خروجی اعمال محاسباتی مثل جمع که به کمک ALU انجام می‌شوند، حتما در ثبات AC ذخیره می‌شود.

AC ← DR + AC
DR ← AC

$d)\ AC \leftarrow TR$

یک کلاک: اگر ثبات AC علاوه بر ALU از روی باس نیز ورودی بگیرد.

AC ← TR

سه کلاک: در صورتی که AC تنها از خروجی ALU تغذیه کند.

O1 ← 0 بلافصل
O2 ← TR
AC ← O1 + O2

$e)\ AC \leftarrow M[AR]$

یک کلاک: اگر ثبات AC علاوه بر ALU از روی باس نیز ورودی بگیرد.

AC ← M[AR]

سه کلاک: در صورتی که AC تنها از خروجی ALU تغذیه کند.

O1 ← 0 بلافصل
O2 ← M[AR]
AC ← O1 + O2

۳. واحد کنترل پردازنده‌ی ۱۶ بیتی بر اساس معماری مجموعه دستورالعمل زیر طراحی شده است.

الف) توضیح دهید معماری مجموعه دستورالعمل، چه اطلاعاتی را در اختیار طراح قرار می‌دهد (حداقل ۳ مورد ذکر شود).

اطلاعاتی که ISA در اختیار ما قرار می‌دهد مجموعه‌ای از دستورالعمل‌هایی است که قرار است واحد کنترل پردازنده‌ی قادر به اجرای آن‌ها باشد. با دانستن این دستورالعمل‌ها می‌توانیم تعداد منابع مورد نیاز از جمله ثبات‌های مورد نیاز در پردازنده و اندازه‌ی دستورالعمل‌ها را متوجه شویم. همچنین تعداد عملوندهای موردنیاز در دستورات و محل ذخیره‌سازی عملوندها (ثبات یا حافظه و غیره) نیز باید در ISA مشخص باشد.

هم‌چنین از روی ISA می‌توانیم متوجه شویم که پردازنده‌ی ما RISC است یا CISC.

ب) اگر این پردازنده، دارای ۳۲ ثبات عام‌منظوره‌ی ۱۶ بیتی باشد، قالب دستورالعمل مناسبی را برای این سیستم طراحی کنید.

MOV <reg1> <reg2>	انتقال محتوای reg2 به reg1 (از هر یک از ثبات‌های عام‌منظوره به یکدیگر)
LDI <reg> <immediate 8-bit>	انتقال محتوای داده‌ی ورودی بلافصل به هر یک از ثبات‌های عام‌منظوره
LOAD <reg> <addr>	انتقال محتوای آدرس مورد نظر به هر یک از ثبات‌های عام‌منظوره
STORE <addr> <reg>	انتقال محتوای ثبات عام‌منظوره‌ی مورد نظر به آدرس حافظه‌ی مورد نظر
ADD <reg> <immediate 8-bit>	به‌روز رسانی محتوای ثبات با مجموع مقدار ثبات و داده‌ی ورودی بلافصل
ADD <reg> <addr>	به‌روز رسانی محتوای ثبات با مجموع مقدار ثبات و آدرس مورد نظر
SUB <src_reg> <dst_reg>	به‌روز رسانی محتوای ثبات مقصد با تفاضل ثبات‌های مبدا و مقصد
PUSH <reg>	انتقال محتوای هر یک از ثبات‌های عام‌منظوره به سر پشته

پردازنده ۱۶ بیتی است ← طول آدرس و کلمه هر دو ۱۶ بیتی

۳۲ ثبات عام منظوره ← حداقل ۵ بیت برای مشخص کردن یک ثبات

۸ دستور داده شده است ← حداقل ۳ بیت برای opcode

اگر حداقل میزان بیت مصرفی هر دستور را مانند جدول زیر بنویسیم، متوجه می‌شویم که دستوراتی مانند LOAD یا STORE در یک کلمه جا نمی‌شوند. بنابراین باید تمامی دستورات را دو کلمه‌ای در نظر بگیریم.

جدول اولیه برای طراحی قالب	opcode operands
MOV <reg1> <reg2>	000 r1r1r1r1r1 r2r2r2r2r2 xxx 3+5+5+3=16bit
LDI <reg> <immediate 8-bit>	001 r1r1r1r1r1 i i i i i i i i 3+5+8=16bit
LOAD <reg> <addr>	010 r1r1r1r1r1 a a a a a a a a a a a a a a a a 3+5+16=24bit
STORE <addr> <reg>	011 r1r1r1r1r1 a a a a a a a a a a a a a a a a 3+5+16=24bit
ADD <reg> <immediate 8-bit>	100 r1r1r1r1r1 i i i i i i i i 3+5+8=16bit
ADD <reg> <addr>	101 r1r1r1r1r1 a a a a a a a a a a a a a a a a 3+5+16=24bit
SUB <src_reg> <dst_reg>	110 r1r1r1r1r1 r2r2r2r2r2 xxx 3+5+5+3=16bit
PUSH <reg>	111 r1r1r1r1r1 xxxxxxxx 3+5+8=16bit

با ۳۲ بیتی در نظر گرفتن دستورات فضای خالی زیادی در هر دستور خواهیم داشت. می‌توانیم از این فضای خالی استفاده کنیم و دستورات را بهینه‌تر طراحی کنیم. کد ثبات‌های عام منظوره را که نمی‌توان کد نشده قرار داد زیرا در این صورت برای هر ثبات به ۳۲ بیت نیاز خواهد بود. اما آپکد را می‌توان به جای ۳ بیت کد شده، با ۸ بیت نمایش داد.
بنابراین قالب دستورالمعل برای این ISA به صورت زیر خواهد بود:

	opcode operands
MOV <reg1> <reg2>	00000001 r1r1r1r1r1 r2r2r2r2r2 xxxxxxx…x 8+5+5+(14)=32bit
LDI <reg> <immediate 8-bit>	00000010 r1r1r1r1r1 i i i i i i i i xxxxxxx…x 8+5+8+(11)=32bit
LOAD <reg> <addr>	00000100 r1r1r1r1r1 a a a a a a a a a a a a a a a a xxx 8+5+16+(3)=32bit
STORE <addr> <reg>	00001000 r1r1r1r1r1 a a a a a a a a a a a a a a a a xxx 8+5+16+(3)=32bit
ADD <reg> <immediate 8-bit>	00010000 r1r1r1r1r1 i i i i i i i i xxxxxxx…x 8+5+8+(11)=32bit
ADD <reg> <addr>	00100000 r1r1r1r1r1 a a a a a a a a a a a a a a a a xxx 8+5+16+(3)=32bit
SUB <src_reg> <dst_reg>	01000000 r1r1r1r1r1 r2r2r2r2r2 xxxxxxx…x 8+5+5+(14)=32bit
PUSH <reg>	10000000 r1r1r1r1r1 xxxxxxx…x 8+5+(19)=32bit

۴. یک کامپیوتر پایه دارای مشخصات زیر است:

- گذرگاه داده و آدرس مشترک ۱۶ بیت
- پردازنده دارای ۳۲ ثبات عام منظوره ۱۶ بیتی است.
- حافظه دارای ۵۱۲ ردیف دو بایتی
- معماری مجموعه دستورالعمل‌های آن مطابق جدول زیر است:

دستورالعمل	عملیات نمادین	توضیح
MOV X,Y	X←Y	محتوای ثبات عام‌منظوره Y را به ثبات X منتقل می‌کند.
ADD X,Y	X← X+Y	یک مقدار از آدرس حافظه‌ی Y را با ثبات عام‌منظوره X جمع و در ثبات X ذخیره می‌کند.
SUB X,Y	X← X-Y	یک مقدار از آدرس حافظه‌ی Y را از ثبات عام‌منظوره X کم می‌کند و در ثبات X ذخیره می‌کند.
ADD X,Y,Z	X←Y+Z	دو مقدارZ وY را از ثبات‌های عام‌منظوره جمع و در ثبات X ذخیره می‌کند.
PUSH X	MEM[SP]← X	مقدار ثبات X را به پشته اضافه می‌کند.
POP X	X← MEM[SP]	مقداری را از پشته برمیدارد و در ثبات عام‌منظوره X ذخیره می‌کند.

حافظه دارای ۵۱۲ ردیف دو بایتی ← طول آدرس ۱۰ بیتی و طول کلمه ۱۶ بیتی
۳۲ ثبات عام منظوره ← حداقل ۵ بیت برای مشخص کردن یک ثبات
۶ دستور داده شده است ← حداقل ۳ بیت برای opcode

الف) قالب دستورالعمل مناسب برای کامپیوتر پایه طراحی کنید.

طولانی‌ترین دستورالعمل ADD X,Y,Z است. زیرا با داشتن ۳ بیت برای آپکد و ۵ بیت برای مشخص کردن آدرس هرکدام از ثبات‌ها، روی هم به حداقل ۱۸ بیت نیاز دارد. در اینصورت دستورها در یک کلمه جا نمی‌شوند و باید تمامی دستورها را دو کلمه‌ای یا ۳۲ بیتی در نظر گرفت.

با ۳۲ بیتی کردن دستورها، مشاهده می‌شود که فضای بسیار زیادی از این قالب خالی می‌ماند. برای استفاده‌ی بهینه‌تر از این فضای خالی می‌توان آپکد را به صورت کد نشده یعنی با استفاده از ۶ بیت، در نظر گرفت.

	opcode operands
MOV X,Y	000001 RxRxRxRxRx RyRyRyRyRy 6+5+5+(16)=32bit
ADD X,Y	000010 RxRxRxRxRx AyAyAyAyAyAyAyAyAyAy 6+5+10+(11)=32bit
SUB X,Y	000100 RxRxRxRxRx AyAyAyAyAyAyAyAyAyAy 6+5+10+(11)=32bit
ADD X,Y,Z	001000 RxRxRxRxRx RyRyRyRyRy RzRzRzRzRz 6+5+5+5+(11)=32bit
PUSH X	010000 RxRxRxRxRx 6+5+(21)=32bit
POP X	100000 RxRxRxRxRx 6+5+(21)=32bit

ب) ریز عملیات لازم برای اجرای دستورات این سیستم را بنویسید.

Instruction Fetch	T0: AR ← PC T1: IR[31:16] ← M[AR], PC++ T2: AR ← PC T3: IR[15:0] ← M[AR], PC++
Instruction Decode	با توجه اینکه آپکد رمزشده نیست، نیازی به این مرحله نداریم.
MOV X,Y	T4.IR[26]: RFAR ← IR[20:16] T5.IR[26]: TMP ← RF[RFAR] T6.IR[26]: RFAR ← IR[25:21] T7.IR[26]: RF[RFAR] ← TMP, SC←0 [31:26]000001 [25:21]RxRxRxRxRx [20:16]RyRyRyRyRy
ADD X,Y	T4.IR[27]: AR ←IR[20:11] T5.IR[27]: O1 ← M[AR] T6.IR[27]: RFAR ←IR[25:21] T7.IR[27]: O2 ← RF[RFAR] T8.IR[27]: AC ← O1 + O2 T9.IR[27]: RF[RFAR] ← AC, SC←0 [31:26]000010 [25:21]RxRxRxRxRx [20:11]AyAyAyAyAyAyAyAyAyAy
SUB X,Y	T4.IR[28]: AR ←IR[20:11] T5.IR[28]: O1 ← M[AR] T6.IR[28]: RFAR ←IR[25:21] T7.IR[28]: O2 ← RF[RFAR] T8.IR[28]: AC ← O2 - O1 T9.IR[28]: RF[RFAR] ← AC, SC←0 [31:26]000100 [25:21]RxRxRxRxRx [20:11]AyAyAyAyAyAyAyAyAyAy
ADD X,Y,Z	T4.IR[29]: RFAR ←IR[25:21] T5.IR[29]: O1 ← RF[RFAR]

	T6.IR[29]: RFAR ←IR[20:16] T7.IR[29]: O2 ← RF[RFAR] T8.IR[29]: AC ← O1 + O2 T9.IR[29]: RFAR ←IR[15:11] T10.IR[29]: RF[RFAR] ← AC, SC←0 [31:26]001000 [25:21]RxRxRxRxRx [20:16]RyRyRyRyRy [15:11]RzRzRzRzRz
PUSH X	T4.IR[30]: AR ← SP T5.IR[30]: RFAR ← IR[25:21] T6.IR[30]: M[AR] ← RF[RFAR], SP--, SC←0 [31:26]010000 [25:21]RxRxRxRxRx
POP X	T4.IR[31]: AR ← SP T5.IR[31]: RFAR ← IR[25:21] T6.IR[31]: RF[RFAR] ← M[AR], SP++, SC←0 [31:26]100000 [25:21]RxRxRxRxRx

ج) منطق واحد کنترل(پایه کنترلی) این سیستم را طراحی کنید.

registers	**Increment** PC = T1 + T3 SP = T6.IR[31] **Decrement** Sp = T6.IR[30] **Load** IR = T1 + T3 PC = 0 AR = T0 + T2 + T4.IR[27] + T4.IR[28] + T4.IR[30] + T4.IR[31] RFAR = T4.IR[26] + T6.IR[26] + T6.IR[27] + T6.IR[28] + T4.IR[29] + T6.IR[29] + T9.IR[29] + T5.IR[30] + T5.IR[31] TMP = T5.IR[26] O1 = T5.IR[27] + T5.IR[28] + T5.IR[29] O2 = T7.IR[27] + T7.IR[28] + T7.IR[29] AC = T8.IR[27] + T8.IR[28] + T8.IR[29]
memory	Write = T6.IR[30] Read = T1 + T3 + T5.IR[27] + T5.IR[28] + T6.IR[31]

	X7	X6	X5	X4	X3	X2	X1	X0	S2 S1 S0
Memory	0	0	0	0	0	0	0	1	0 0 0
RFAR	0	0	0	0	0	0	1	0	0 0 1
PC	0	0	0	0	0	1	0	0	0 1 0
SP	0	0	0	0	1	0	0	0	0 1 1
IR	0	0	0	1	0	0	0	0	1 0 0
TMP	0	0	1	0	0	0	0	0	1 0 1
AC	0	1	0	0	0	0	0	0	1 1 0

X7 = 0
X6 = T10.IR[29] + T9.IR[28] + T9.IR[27]
X5 = T7.IR[26]

X4 =T4.IR[26] + T6.IR[26] + T4.IR[27] + T6.IR[27] + T4.IR[28] + T6.IR[28] + T4.IR[29] + T6.IR[29] + T5.IR[30] + T5.IR[31]
X3 = T4.IR[30] + T4.IR[31]
X2 =T0 + T2
X1 = T5.IR[26] + T7.IR[27] + T7.IR[28] + T5.IR[29] + T7.IR[29] + T6.IR[30]
X0 = T1 + T3 + T5.IR[27] + T5.IR[28] + T6.IR[31]

برای طراحی پایه‌ی کنترلی ALU به حداقل دو پایه نیاز خواهیم داشت زیرا ۳ دستور داریم که با ALU سر و کار دارند. هم می‌توان چهار پایه‌ی Y0 تا Y3 را به عنوان پایه‌های ALU در نظر گرفت، هم می‌توان دو پایه در نظر گرفت و آن‌ها را به کمک یک Encoder و Y0 تا Y3 برنامه ریزی کرد. در جدول زیر راه‌حل دوم انتخاب شده است و پایه‌های کنترلی ALU، پایه‌های C0 و C1 هستند.

طراحی پایه‌های کامند ALU	Y3	Y2	Y1	Y0	C1 C0
ADD X,Y	0	0	0	1	0 0
SUB X,Y	0	0	1	0	0 1
ADD X,Y,Z	0	1	0	0	1 0
idle	1	0	0	0	1 1

Y3 = not (T8.IR[29] + T8.IR[28] + T8.IR[27])
Y2 = T8.IR[29]
Y1 = T8.IR[28]
Y0 = T8.IR[27]

لطفا نکات زیر را در نظر بگیرید.

اشکالات خود را می‌توانید از طریق ایمیل autcafall2021@gmail.com بپرسید.

لینک کانال تلگرام درس https://t.me/cafall2021 است. برای اطلاع از اخبار درس دنبال کنید.

موفق باشید