حل مسائل فصل ۵

کلاس حل تمرین معماری کامپیوتر دکتر زینالی، رامتین کوثری، ۱ آذر ۱۴۰۳ کلاس ۳۱۰

طریقه حل مسائل با ۲ یا چند دستور در هر خانه حافظه :

در سوال های امتحان معمولا چند دستور در هر خانه حافظه قرار دارند که ما باید آنها را به درستی اجرا کنیم. برای مثال فرض کنید که هر خانه حافظه دارای ۲ دستور باشد، مانند :

I_1	$Opcode_1$	$Address_1$	I_2	$Opcode_2$	$Address_2$
1	5	10	1	5	10

در این نوع از فرمت دستور ما باید این مسئله را با استفاده از یک فلگ کمکی حل کنیم، حال پیش از آنکه این فلگ را طراحی کنیم باید ابتدا خود فرمت دستور را بررسی کنیم. فرمت دستور ما مجموعا ۳۲ بیت است که بخش سمت چپی دستور اول و بخش سمت راستی دستور دوم میباشد. اگر ما دستور بخش اول را انجام دهیم باید به بخش دوم رفته و آن را انجام دهیم ولی حواستان باشد که از مراحل چرخه دستورات مرحله Fetch را برای بخش های دوم انجام نمی دهیم چون Fetch می آید و از حافظه یک بلوک را میخواند، حال که بلوک حافظه ما از ۲ دستور تشکیل شده است هنگامی که دستور اول را خواندیم در همان بلوک Fetch شده باید به سراغ دستور دوم در بخش دوم برویم و نباید Fetch کنیم. حال اگر در بخش دوم باشیم باید برویم بخش اول بلوک بعدی حافظه پس اینجاست که دوباره باید Fetch را انجام دهیم زیرا بلوک بعدی را میخواهیم. تا اینجا در نظر داشته باشید که ما برای تعیین بودن در بخش اول و بخش دوم نیاز داریم که یک فلگ کمکی یا جانبی داشته باشیم و SC را فقط هنگامی که در بخش دوم (یا بخش آخر در حالتی که چندین دستور داشته باشیم) باید ریست (صفر) کنیم. این نکته را هم توجه داشته باشید که مرحله بعد از Fetch مرحله Decode میباشد و در این مرحله اجزای دستور را تفکیک می کردیم، حال اگر از بخش اول گذر کنیم باید برای بخش دوم نیز این کار را انجام دهیم و همچنین مراحل بعدی مانند E.A و Execute. پس برای اینکه مرحله Fetch را انجام ندهیم ولی مرحله Decode را انجام دهیم باید Sequencer را به روی زمانی ببریم که مرحله Decode شروع میشود.

به حل مثال زیر از امتحان دقت کنید که میخواهیم مراحل چرخه اجرای دستورات را برای این حالت بنویسیم :

در این مثال ابتدا باید آن فلگ کمکی را تعین کنیم، ما ۲ دستور در یک بلوک بیشتر نداریم پس با ۱ بیت میتوانیم آن را کد کنیم که مثلا با m

نمایش میدهیم حال اگر $m' = 0$: m **باشد قرارداد می کنیم بخش اول**

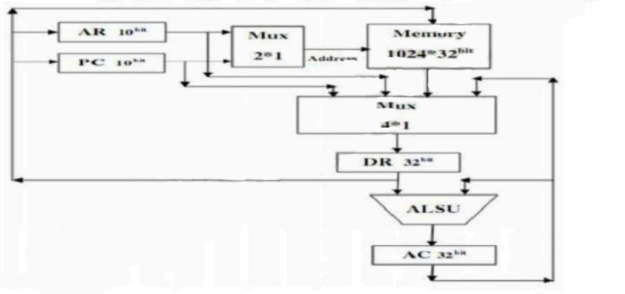
و اگر $m = 1$: m **باشد قرارداد می کنیم بخش دوم** اجرا شود :

m' m

I_1	$Opcode_1$	$Address_1$	I_2	$Opcode_2$	$Address_2$
1	5	10	1	5	10

حال با این قرارداد می توانیم مسئله را حل کنیم. مراحل چرخه اجرای دستورات به شکل زی

۴- واحد DPU(Data Path Unit) و نوع وتعدادی از دستورات یک پردازنده بصورت زیراست:



I_1	$Opcode_1$	$Address_1$	I_2	$Opcode_2$	$Address_2$
1^{bit}	5^{bit}	10^{bit}	1^{bit}	5^{bit}	10^{bit}

(I = 0 Indirect , 1 Direct)

این مرحله چون ۱ بلوک حافظه را در مقدار PC میخواند پس نیازی نداریم که فلگ جانبی را لحاظ کنیم و از آنجایی که شکل رجیستر IR ندارد باید از رجیستر DR استفاده کنیم، همچنین PC مستقیم به آدرس مموری متصل است پس خواهیم داشت :

$$T_0 : \quad DR \leftarrow M[PC] \ , \ PC \leftarrow PC + 1$$

البته توجه کنید اگر DR ما ۱۶ بیت یعنی نصف اندازه خانه حافظه بود یکبار بخش اول و بار دیگر بخش دوم را داخل DR بنویسیم. یعنی همان زمانبندی نوشته شده منتهی در ۲ خط با نوشتن شروط m و m'.

Decode : $I \leftarrow IR_{(15)}\ ,\ D_0D_1D_2 \dots D_7 \leftarrow IR_{(14-12)}\ ,\ AR \leftarrow IR_{(11-0)}$

این مرحله باید حواسمان باشد که یکبار باید برای m' و بار دیگر باید برای m نوشته شود یعنی :

$m'T_1 : \quad I \leftarrow DR_{(15)}\ ,\ D_0D_1D_2 \dots D_{31} \leftarrow IR_{(14-10)}\ ,\ AR \leftarrow IR_{(9-0)}$

$m\,T_1 : \quad I \leftarrow DR_{(31)}\ ,\ D_0D_1D_2 \dots D_{31} \leftarrow IR_{(30-26)}\ ,\ AR \leftarrow IR_{(25-16)}$

Effective Address : $if\ (I = 1)\ then\ AR \leftarrow M[AR]$

در این مرحله چون در مرحله قبل چه آدرس دستور بخش اول و چه آدرس بخش دوم را در AR ریختیم پس عملا نباید رفتار متفاوتی به کمک فلگ جانبی داشته باشیم زیرا قرار است حافظه در AR به AR ریخته شود و AR در مرحله قبل یا آدرس نیمه اول را دارد یا دوم :

$I\,T_2 : \quad NOP$

$m\,I'T_2 + m'I'T_2 : \quad DR \leftarrow M[AR]$

$m\,I'T_3 + m'I'T_3 : \quad AR \leftarrow DR$

با توجه به نکته گفته شده میتوانیم از فلگ جانبی فاکتور بگیریم :

$I\,T_2 : \quad NOP$

$I'T_2(m + m') \Rightarrow I'T_2(1) \Rightarrow I'T_2 : \quad DR \leftarrow M[AR]$

$I'T_3(m + m') \Rightarrow I'T_3(1) \Rightarrow I'T_3 : \quad AR \leftarrow DR$

Execute : $D_0D_1D_2 \dots D_{31}$

برای **مثال** مرحله اجرا دستور ADD را می نویسیم :

$D_{23}(ADD) : \quad M[AR] \leftarrow AC + M[AR]$

$D_{23}T_4 : \quad DR \leftarrow M[AR]$

$D_{23}T_5 : \quad AC \leftarrow AC + DR$

$D_{23}T_6 : \quad DR \leftarrow AC$

$D_{23}T_7 : \quad M[AR] \leftarrow DR$

حال در اینجا دستور تمام شده و باید سراغ دستور بعدی برویم ولی حواستان باشد که دستور بعد در خانه بعدی حافظه نیست پس نباید SC را صفر کنیم تا مرحله Fetch اتفاق بیفتد بلکه فقط باید به بخش دوم ($m \leftarrow 1$) برویم و همچنین SC را ۱ کنیم تا چرخه از مرحله Decode در دستور بخش دوم شروع شود و وقتی دستور بخش دوم تمام شد SC باید صفر شود و همچنین فلگ جانبی را نیز باید دوباره صفر کنیم ($m \leftarrow 0$) که چرخه اجرای دستور بعدی از بخش اول در خانه بعدی حافظه شروع شود پس کافی است به RTL های بالا، RTL های **جدیدی** اضافه کنیم که این فرایند را انجام دهد :

$D_{23}T_4 : \quad DR \leftarrow M[AR]$

$D_{23}T_5 : \quad AC \leftarrow AC + DR$

$D_{23}T_6 : \quad DR \leftarrow AC$

$D_{23}T_7 : \quad M[AR] \leftarrow DR$

$m'D_{23}T_7 : \quad SC \leftarrow 1\ ,\ m \leftarrow 1$

$m\,D_{23}T_7 : \quad SC \leftarrow 0\ ,\ m \leftarrow 0$

پس میتوانیم به کمک فلگ جانبی این فرایند ها را انجام دهیم ولی باید توجه داشته باشیم که به ازای هر فلگی که اضافه میکنیم باید یک $J.K - f.f$ (جی کا فلیپ فلاپ) برای آن طراحی کنیم.