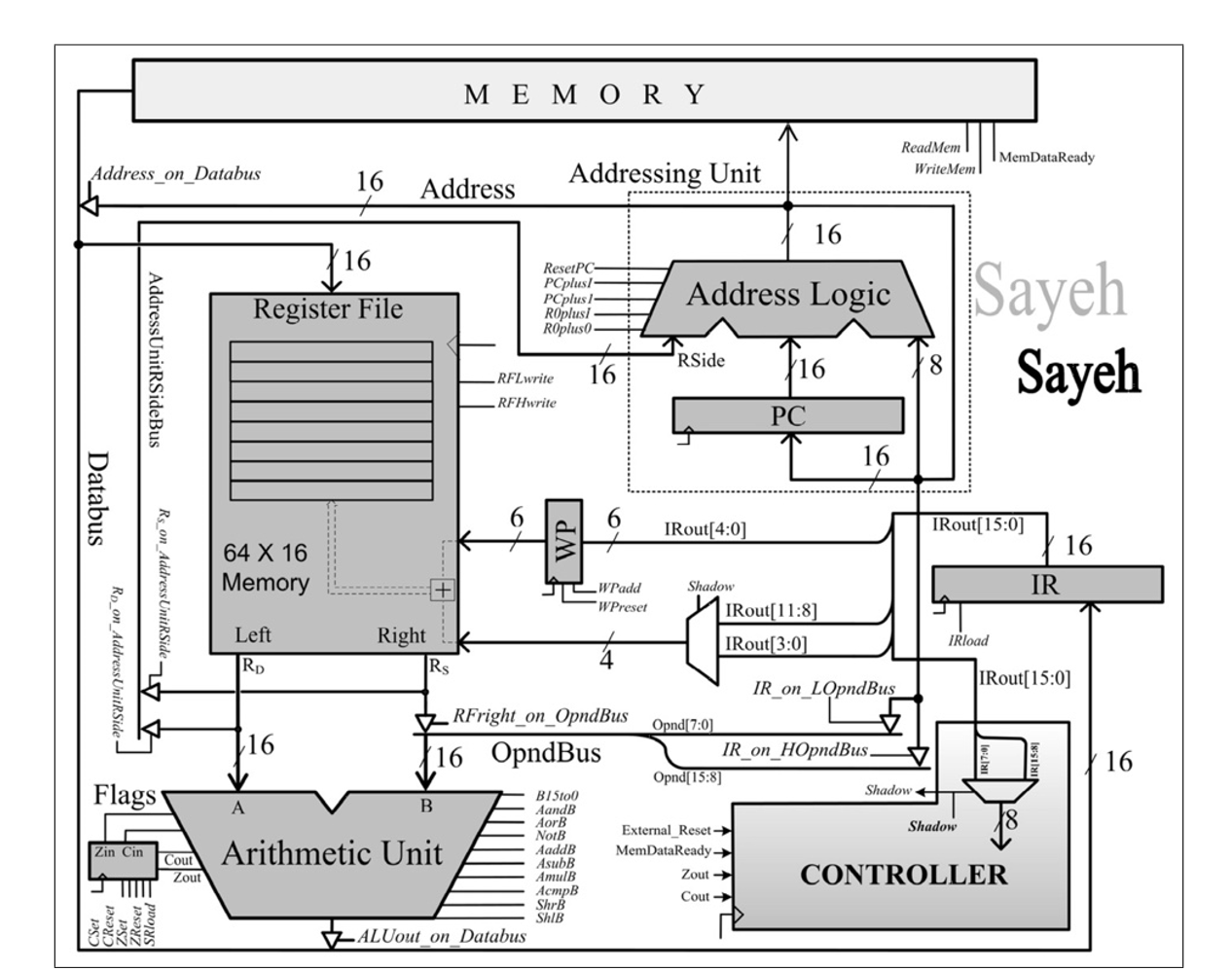
به نام خداوند بخشنده و مهربان

گزارش پروژه معماری کامپیوتر- طراحی کامپیوتر پایه SAYEH با زبان VHDL

این کامپیوتر بر طبق شکل زیر طراحی شده و دارای کامپوننت های امده در شکل است.



دستورات در مموری قرار گرفته و از ان جا به داخل IR امده و دیکد می شوند. کنترلر وظیفه دیکد و بر اساس ان کنترل سیگنال های کنترلی بقیه بخش ها را دارد. در datapath هم ماژول ها به هم وصل شده و کنترل بافر های سه حالته در این بخش صورت می گیرد. به طور مثال در صورتی که سیگنال address\_on\_databus یک باشد سیم خروجی address\_logic به سیم databus متصل می شود.

Reset :

برای شروع کار باید ابتدا کنترلر را ریست کنیم. برای این کار مقدار سیگنال controller\_rst بعد از 0 بودن 1 کنیم تا کنترلر ریست به استیت ریست شود.( 0 سپس یک زیرا حساس به لبه بالارونده این سیگنال است و اگر فقط یک باشد این اتفاق صورت نمی گیرد). در این مرحله pc و wp هر کدام مقدار های پیش فرض خود که 0 هست را می گیرند( سیگنال pcreset ورودی address\_logic هست و مقدار خروجی ان را 0 می کند. از ان جایی که خروجی address\_logic ورودی pc هست با یک کردن pcenable با امدن کلاک مقدار 0 در خروجی pc هم می اید و با این راه مقدار pc 0 می شود) و همین طور flag های c و z نیز مقدار 0 به خود می گیرند. همین طور کنترلر برای fetch کردن دستور بعدی اماده می شود. (next\_state<= fetch) تغییر استیت حساس به لبه پایین رونده می باشد. (در لبه پایین رونده current\_state<=next\_state )

Fetch :

در این مرحله می خواهیم دستور را از ادرس مورد نظر از مموری خوانده و ان را وارد IR کنیم. محاسبه ادرس مورد نظر توسط ماژول address\_logic صورت می گیرد که در ابتدا مقدار ان 0 است. در این استیت این روند را داریم:

1. در لبه پایین رونده کلاک وارد این استیت شده و مقدار سیگنال های reamem و IRload یک می شود. تا دستور از مموری خوانده و به IR برود. مقدار shadow هم 1 می شود که نشان دهد ابتدا 8 بیت پرارزش IR دیکد شود. استیت بعدی decode در نظر گرفته می شود. مقدار یکسری سیگنال هم 0 می شود این برای این است که زمانی که پس از اجرای دستور هایی وارد استیت fetch می شویم همه ان سیگنال هایی که برای اجرای ان دستور یک شده بودند در این جا 0 شوند.
2. در لبه بالا رونده دستور از مموری به databus می اید (مموری حساس به لبه بالارونده است)
3. در لبه پایین رونده دستور از databus که ورودی IR هست به خروجی ان رفته و استیت هم به decode تغییر می کند. (IR حساس به لبه پایی رونده هست)

توضیح: در طول کار ابتدا IR هم حساس به لبه بالا رونده بود ولی به مشکل برخورد کردم چرا که درست همان موقع که دستور از مموری وارد databus می شد IR هم می خواست ان را در خروجی خود قرار دهد و این باعث مشکل می شد. برای همین IR را حساس به لبه پایین رونده کردم تا ابتدا دستور به databus بیاید و بعد از ان بخواهد وارد IR بشود و به این صورت مشکل حل شد.

Decode:

در لبه پایین رونده وارد این استیت می شویم و سیگنال های IRload و readmem 0 می شوند. تا مقدار دستور در IR مانده و دستور دیگری از مموری خوانده نشود. Instruction ان بخش از دستور 16 بیتی هست که می خواهیم ان را دیکد کنیم. مقدار ان نیز بر حسب مقدار shadow تعیین می شود. در صورت 1 بودن shadow 8 بیت پر ارزش IR و در صورت 0 بودن ان 8 بیت کم ارزش.

اصل عملیات دیکد در لبه بالا رونده بعدی صورت می گیرد. به این علت که دقیقا همان لحظه ای که وارد دیکد می شویم خروجی IR هم می اید و نمی توان مقدار ان را در همان لحظه چک کرد. برای همین به انداره نیم کلاک صبر کرده و بعد عملیات دیکد را انجام می دهیم.

در ادامه توضیح می دهیم که برای اجرا هر دستور چه مراحلی صورت می گیرد:

Nop:

برای این دستور لازم نیست کاری انجام شود فقط باید استیت بعدی مشخص شود و خود به خود به اندازه 1 کلاک کاری انجام نشده. (از ورود به دیکد ت خروج از ان) برای تشخیص استیت بعدی به روش زیر عمل می کنیم:

چون این دستور immediate ندارد ممکن است دارای shadow باشد. پس دو حالت ممکن است اتفاق بیفتد:

1. مقدار shadow الان 0 است که به این معنی است که دستور در 8 بیت کم ارزش IR بوده پس در مرحله بعدی باید دوباره از مموری fetch کنیم. پس استیت بعدی fetch در نظر گرفته می شود.
2. مقدار shadow در حال حاضر 1 است که به این معنی است که این دستور در 8 بیت پر ارزش قرار داشته و باید در مرحله بعدی 8 بیت کم ارزش دیکد شود. در این صورت مقدار shadow را 0 کرده و استیت بعدی را decode می گذاریم. تا در کلاک بعدی بخش کم ارزش IR دیکد شود. (از ان جایی که process ما در این جا حساس به کلاک و current\_state هست تغییر ندادن استیت مشکلی ایجاد نمی کند و در لبه پایین رونده بعدی دوباره وارد دیکد شده و این بار به خاطر 0 بودن shadow 8 بیت کم ارزش دیکد شوند) این روند را برای همه دستور های 8بیتی(بدون immediate ) ممکن است داشته باشیم.

Halt :

برای اجرای این دستور بعد از استیت دیکد به استیت هالت می رویم و تا زمانی که extrenal\_reset کنترلر فعال نشده باشد در همین استیت می مانیم. در صورت 1 شدن external\_reset همان روندی را که برای دستور های 8 بیتی داشته ایم تکرار می کنیم. یا به استیت fetch و یا به استیت fetch2 می رویم.(رجوع شود به بخش nop )

C and Z set or reset :

برای مثلا ست کردن فلگ C سیگنال cset را 1 می کنیم و بعد از ان به استیت fetch یا fetch2 می رویم. به صورت مشابه برای ریست کردن و ست کردن هر دو فلگ عمل می کنیم.

Clear wp :

برای اجرای این دستور نیز سیگنال wpreset را 1 می کنیم و به استیت fetch یا fetch2 می رویم.

Move register :

برای اجرای این دستور باید مقدار rs را به databus برسانیم. برای این کار سیگنال rfonopndbus را یک می کنیم تا مقدار rs وارد alu شود. و همین طور سیگنال bout را هم یک کرده تا alu مقدار ورودی خود را مستقیما به خروجی انتقال دهد. استیت بعدی را exe قرار می دهیم.

در استیت exe، aluondatabus را یک می کنیم تا خروجی alu روی databus قرار بگیرد. (توضیح: مشکلی که در این قسمت به ان برخوردم این بود که اول در همان استیت decode مقدار aluondatabus را هم یک می کردم ولی چون همان لحظه مقدار خروجی alu عوض شده بود مقدار ان روی databus ریخته نمی شد. برای همین در این استیت مقدار aluondatabus را یک کردم تا کمی قبل خروجی alu اماده شده باشد بعد بخواهد مقدار ان را روی databus بنویسد) همین طور مقدار rfhwrite و rflwrite را هم یک کرده تا register file اماده نوشتن شود و در لبه بالارونده بعدی دیتا نوشته می شود. استیت بعدی fetch یا fetch2 در نظر گرفته می شود و در لبه پایین رونده بعدی استیت تغییر می کند.

Load addressed :

برای اجرای این دستور ابتدا باید مقدار rs را به عنوان ادرس به مموری بدهیم. برای این کار rsonaddressbus را یک می کنیم تا مقدار rs به ورودی address\_logic برود. سیگنال rplus0 را هم یک می کنیم تا address\_logic مقدار rside خود را که در این جا همان rs شده مستقیما به عنوان ادرس به مموری بدهد. سیگنال های IRload را هم 0 می کنیم تا دیتای خوانده شده وادر IR نشود و همین طور مقدار pcen را هم 0 می کنیم ت مقدار قبلی pc نگه داشته شود و خروجی address\_logic که همان rs است وارد pc نشود تا بتوانیم بعد از اجرای این دستور به دستور بعدی برویم. استیت بعدی exe در نظر گرفته می شود.

در استیت exe مقدار readmem را یک می کنیم تا در لبه بالارونده از مموری خوانده شود. در لبه پایین رونده مقدار های rflwrite و rfhwrite را یک می کنیم تا registerfile اماده نوشتن باشد و استیت بعدی exe2 می باشد. بعد از ورود به این استیت در لبه بالا رونده مقدار دیتا موجود روی databus به روی registerfile نوشته می شود. و به استیت fetch و یا fetch2 می رویم.

Store addressed :

ابتدا باید مقدار rd را به عنوان ادرس به مموری بدهیم. پس rdonrside و rplus0 را یک می کنیم. از طرفی دیتای موجود در rs را هم باید به databus برسانیم. پس rfonopndbus و bout را یک می کنیم. (البته در این استیت دیتا هنوز وارد databus نشده است). مقدار سیگنال های pcen و IRload را هم 0 می کنیم تا خروجی address\_logic وارد pc نشود و مقدار روی databus روی IR نوشته نشود. استیت بعدی را exe می گیریم.

در لبه پایین رونده وارد exe می شویم در لبه بالارونده مقدار سیگنال aluondatabus را یک می کنیم تا خروجی alu که الان همان مقدار rs است وارد databus شود. Writemem را هم یک می کنیم تا مموری اماده نوشتن شود. چون نیاز به یک لبه بالارونده دیگر داریم استیت بعدی را exe2 در نظر می گیریم و پس از ان به fetch یا fetch2 می رویم.

دستور های alu :

سیگنال rfonopndbus را 1 می کنیم تا مقدار rs ورودی alu قرار بگیرد. استیت بعدی را exe قرار می دهیم. در exe سیگنال عملیات مربوطه را 1 می کنیم. در لبه بالا رونده aluondatabus را یک می کنیم تا خروجی alu روی databus قرار بگیرد. اگر در همان لبه پایین رونده قبلی این کار را می کردیم چون خروجی alu همان موقع امده ان مقدار روی databus نوشته نمی شد برای همین به اندازه نیم کلاک صبر کرده و بعد خروجی alu را databus می نویسیم. همین طور rfhwrite و rflwrite را نیز 1 کرده تا registerfile اماده نوشتن شود. چون به یک لبه بالارونده دیگر برای نوشته شدن در Registerfile نیاز داریم استیت بعدی را exe2 قرار می دهیم و بعد از ان به fetch یا fetch2 می رویم.

Mil و mih :

مقدار irlonopndbus را یک می کنیم تا مقدار immediate در ورودی alu قرار بگیرد و همین طور مقدار bout را هم یک می کنیم تا همین مقدار ورودی در خروجی alu قرار بگیرد. rflwrite را هم یک می کنیم تا registerfile اماده نوشتن شود. استیت بعدی را exe قرار می دهیم

در exe مقدار aluondatabus را یک می کنیم تا خروجی alu روی databus قرار بگیرد(همان مقدار immediate الان روی خروجی alu قرار دارد). چون این دستور 16 بیتی است استیت بعدی fetch در نظر گرفته می شود. توجه شود که هر بار می خواهیم به استیت fetch برویم سیگنال pcplus1 را یک می کنیم ت مقدار pc یکی افزایش یابد و همین طور سیگنال های IRload و pcen را هم یک می کنیم تا اگر در مراحل قبلی 0 شده بودند این جا حتما 1 شوند که دستور خوانده شده از مموری حتما به داخل IR برود و مقدار افزایش یافته pc هم وارد pc شود.

Save pc :

rfhwrite<='1'; rflwrite<='1'; dataonaddressbus<='1';

next\_state<=exe2;

سیگنال pcplusI 1 می شود ت مقدار pc با مقدار immediate در address\_logic جمع شود و مقدار pcen 0 می شود تا این حاصل جمع وارد pc نشود و pc مقدار قبلی خود را حفظ کند. استیت بعدی exe در نظر گرفته می شود.

در exe مقدار addressondatabus 1 می شود ت خروجی address\_logic به روی databus بیاید. از طرفی مقادیر rflwrite و rfhwrite را هم 1 می کنیم تا در لبه بالا رونده بعدی دیتا به روی registerfile نوشته شود. استیت بعدی را exe2 در نظر می گیریم و بعد از ان به fetch می رویم و سیگمال های گفته شده را نیز فعال می کنیم.

Jump addressed :

rdonrside را 1 می کنیم تا مقدار rd ورودی address\_logic. قرار بگیرد. استیت بعدی exe

در exe مقدار rplusI را یک می کنیم تا حاصل جمع در خروجی address\_logic قرار گرفته و همین ظور pcen را هم یک می کنیم تا در کلاک بعدی این مقدار وارد pc شود.

Jump relative :

pcplusI را 1 می کنیم. استیت بعدی Fetch

Add wp :

Wpadd را یک می کنیم. استیت بعدی exe و بعد از ان وارد fetch می شویم.