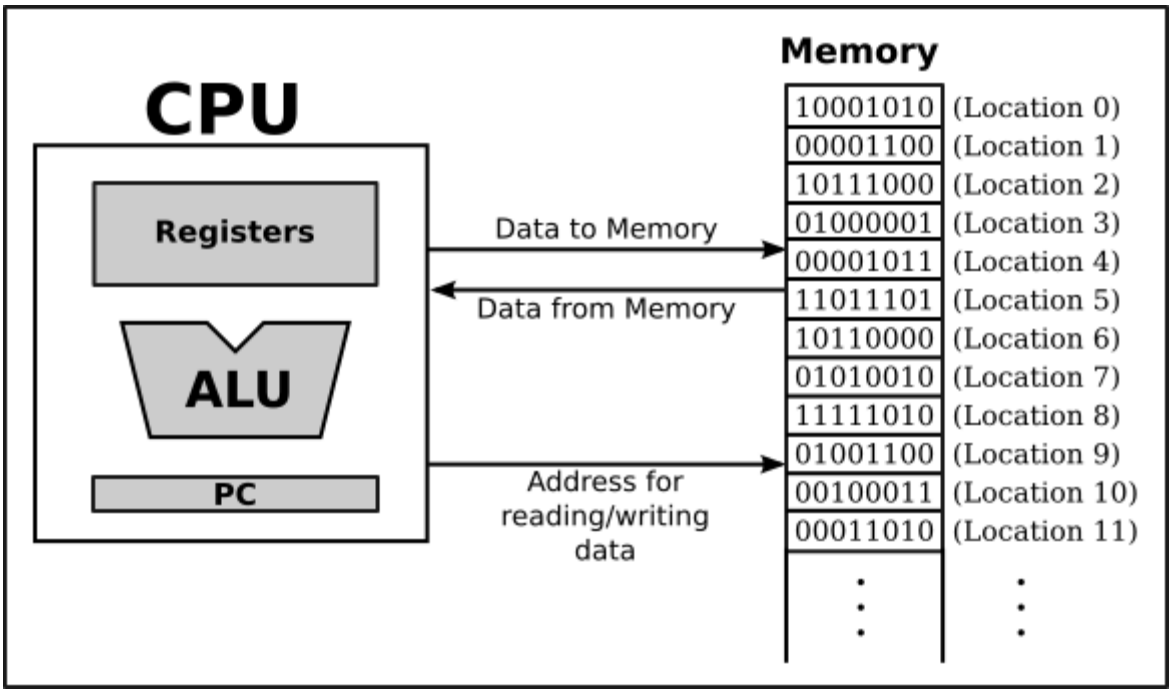
آزمایش یازدهم: کامپیوتر پایه

در این آزمایش قصد داریم درمورد کامپیوتر های پایه صحبت کنیم. به طور کلی کامپیوتر پایه از مجموعه ای از رجیستر ها، کلاک و مرکز کنترل و یکسری Instruction (دستورالعمل) تشکیل شده است. هدف طراحی یک کامپیوتر دیجیتال انجام میکرو عملیات های مختلف است به گونه ای که میتوانیم آن را دستور دهی کنیم به شکلی که عملیات ها را باترتیبی خاص اجرا نماید (برنامه نویسی).

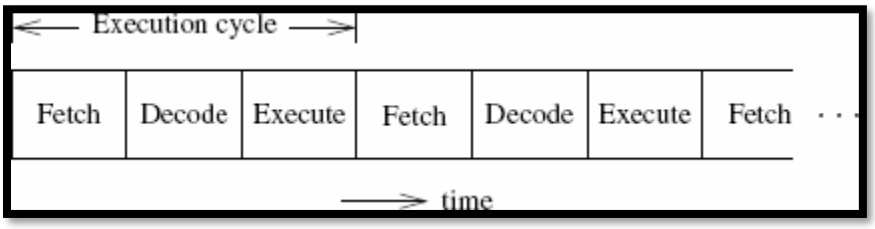
در طراحی کامپیوتر پایه برای ساده سازی صرفا دارای یک مموری اصلی و یک پردازنده هستیم که پردازنده شامل بخش های ALU , Control Unit است.

در این گزارش برای ساده سازی به IO نمیپردازیم.

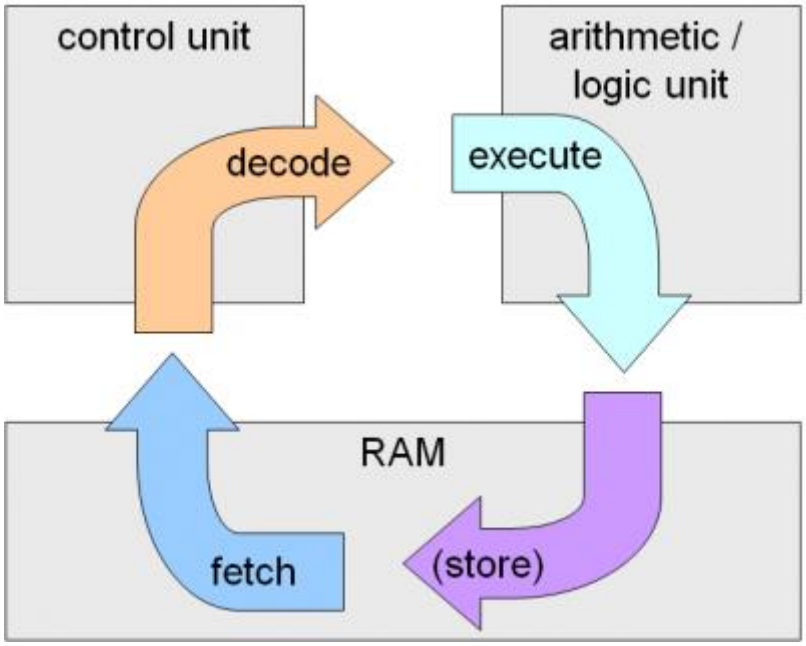


چرخه دستورالعمل ها

دستورالعمل تحت نظر Control Unit (مرکز کنترل) مدیریت میشوند. این چرخه برای هر دستورالعمل تکرار میشود که دارای ۶ فاز مختلف است.



- فاز اول: Fetch Instruction
- فاز دوم: Decode Instruction
- فاز سوم: Fetch Operands
- فاز چهارم: Execute
- فاز پنجم: Store Result

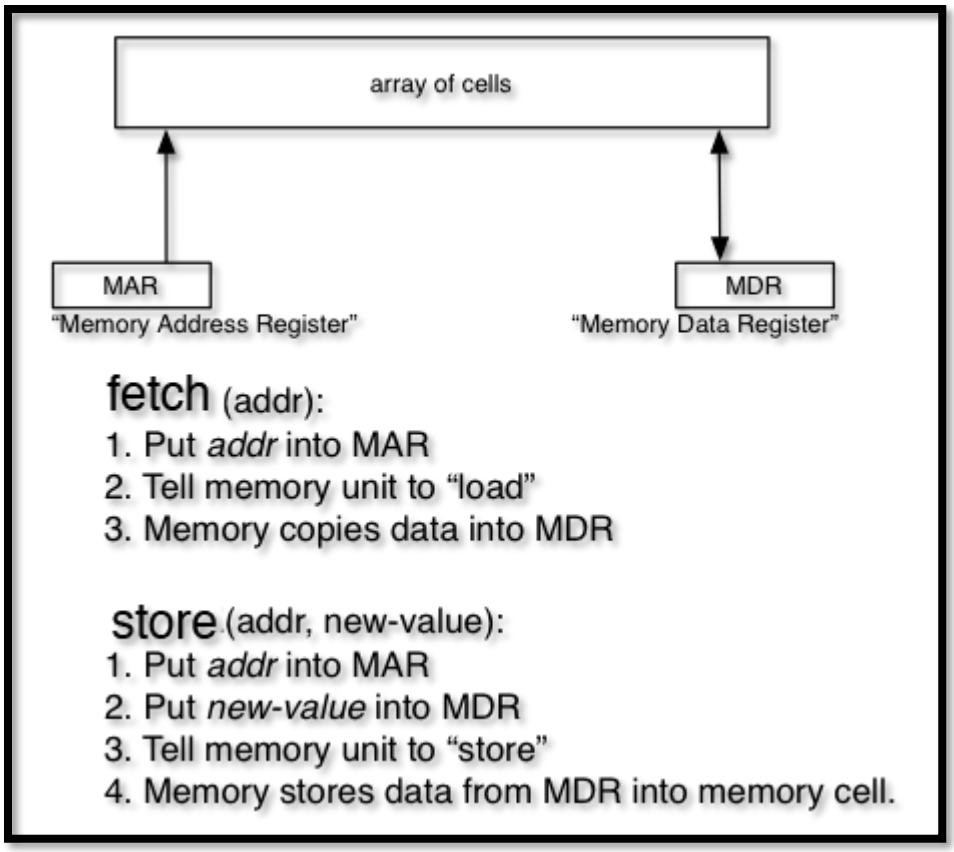


فاز اول

- دیتای instruction را از مموری دریافت میکنیم
- آن را در رجیستر IR لود میکنیم
- AR (Address Register) همزمان با آن لود میشود
- DR (Data Register) نیز به همین شکل

- PC (یا همان Program Counter) را یک عدد افزایش میدهد که در چرخه بعدی سراغ Instruction بعدی برویم

در شکل زیر به جای AR از MAR و به جای DR از MDR استفاده شده است.

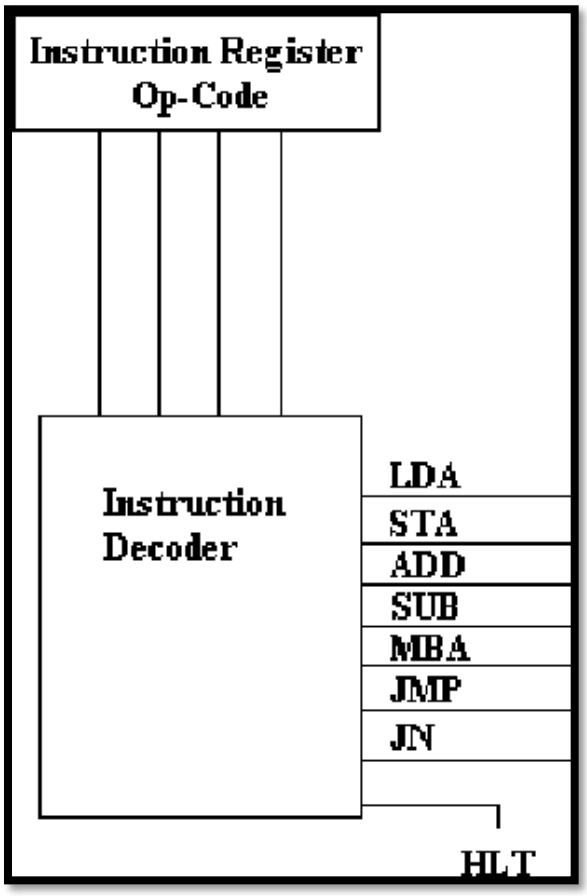


همانطور که در شکل فوق پیداست، دو حالت در این بخش باید هندل کنیم، اولین حالت، حالت Fetch و دومی Store است. یعنی هم بتوانیم یک آدرس حافظه را بخوانیم (برای مثال یک دیتا یا یک Instruction) و همچنین بتوانیم روی موری بنویسیم. (رابطه بین CPU & MM)

فاز دوم: دکود کردن دستورالعمل

فرایند دکود کردن به CPU این امکان را میدهد تا تعیین کند چه دستورالعمل هایی باید اجرا شود تا CPU بتواند برای انجام دستورالعمل عمل وند ها را Fetch کند(درواقع بیس Micro Operation ما تعیین میکنیم که مدل Fetch عملوند ها چطور باشد). OPCODE گرفته شده از حافظه برای مراحل بعد Decode میشود و به رجیستر های موردنیاز منتقل میشود.

خروجی این فاز یک Output line یکتاست که یک عملیات به خصوص را انجام میدهد.



این Micro Operation ها هنگام طراحی پردازنده، طراحی میشوند و قرار میگیرند. دربالا یک نمونه ساده را شاهد هستیم.

فاز سوم: خواندن عملوند(Operand fetch)

در این فاز ابتدا بررسی میکنیم که دستور العمل حافظه ای هست یا نه. اگر دستور العمل حافظه ای بود آنگاه آدرس مموری که در دستورالعمل موجود است را به AR میدهیم. زیرا AR دسترسی مستقیم به مموری را دارد. سپس read مربوط به مموری 1 میشود تا بتوانیم به وسیله یARِ از مموری، داده را بخوانیم و سپس داده را در DR(data register) ذخیره میکنیم. به این ترتیب علمیات خواندن عملوند را انجام دادیم. در این مرحله به دنبال آدرس موثر هستیم؛ اگر که آدرس به صورت مستقیم بود؛ این آدرس همان آدرس موثر میباشد و اگر که غیر مستقیم بود یکبار داده را که محتوی آدرس است از مموری میخوانیم و در AR ذخیره میکنیم. این آدرس

جدید همان آدرس موثر میباشد که به داده اشاره میکند. پس به کمک این آدرس که در AR ذخیره شده است مجددا داده را از مموری میخوانیم و در DR ذخیره میکنیم.

اگر دستورالعمل ثباتی بود نیاز به خواندن operand نیست.

فاز چهارم: اجرای دستور العمل

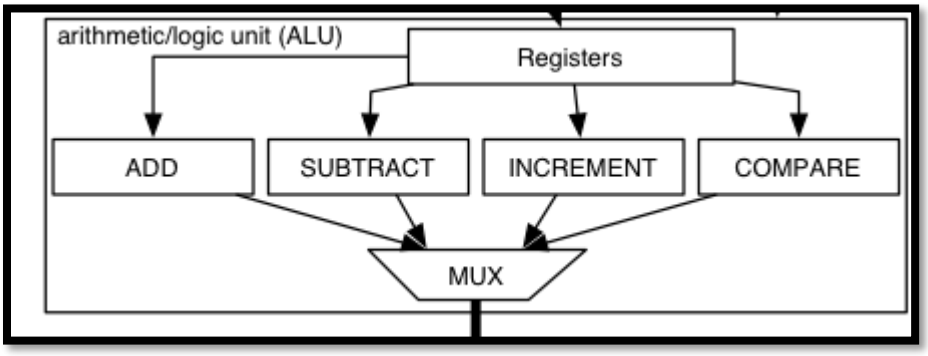
در این فاز که پس از خوانده شدن عملوند است دستورالعمل را اجرا میکنیم. با توجه به opcode که decode شده است نوع عملیات مشخص میشود.

یک سری ریزعملیات داریم ((micro operation (MO)که با توجه به اینکه شرط کدام یک از ریزعملیات ها درست است(با توجه به opcode که دیکود شده)؛ آن ریزعملیات اجرا میشود.

مثال: در یک حالت خاص که قرار است دو عملوند که در حافظه هستند را با هم جمع کنیم مراحل به صورت زیر است:

۱- ابتدا آدرس عملوند اول را از IR میخوانیم و درون AR ذخیره میکنیم
۲- سپس به وسیله ی AR داده را از مموری میخوانیم و در یک ثبات مثل O1 ذخیره میکنیم.
۳- این سری آدرس دوم را از IR میخوانیم و درون AR ذخیره میکنیم.
۴- سپس به وسیله AR داده را از مموری میخوانیم و در یک ثبات مثل O2 ذخیره میکنیم
۵- عملیات جمع را انجام میدهیم و مقئار را در O3 ذخیره میکنیم. (از طریق ALU)
۶- مقدار موجود در O3 را در خانه ای از حافظه که آدرس آن AR هست(عملوند دوم) قرار میدهیم.

توضیح بالا در واقعا ریزعملیات مربوط به جمع ۲ عملوند بود که نتیجه در عملوند دوم ذخیره شد.



BUS:

کامپیوتر پایه داری ۸ رجیستر، Memory Unit و Control Unit است.برای انتقال داده ها بین رجیستر ها و موری و مرکز کنترل نیازمند یک مسیر بهینه هستیم. برای اینکار از Common BUS استفاده میکنیم.

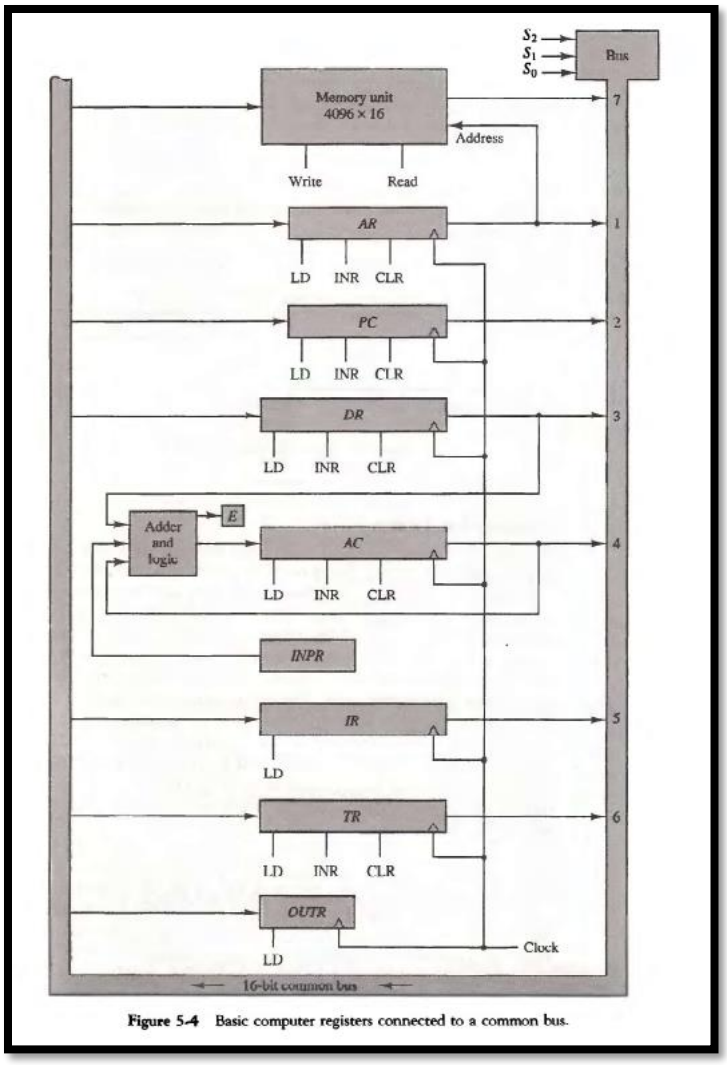


Figure 5-4 Basic computer registers connected to a common bus.

با توجه به شکل فوق اگر بخواهیم برای مثال دیتای DR را بخوانیم به عدد دسیمال کنار آن دقت میکنیم، برای مثال نوشته شده است ۳. یعنی معادل باینری آن S2S1S0 = 011 که براساس آن میتوانیم بفهمیم کدام لاین فعال است و دیتای کدام رجیستر داریم میخوانیم.

موفق باشید