ویژگی های RISC

- دستورات ساده و کمی دارند
  - هدف این است که بتوان دستورات را بسرعت اجرا کرد .اغلب دستورات RISC در یک سیکل اجرا میشوند (بعد ازواکشی و رمزگشایی)
  - چون دستورات در زمان مشابهی اجرا میشوند عمل pipeline دارای بازده بالایی خواهد بود.
  - کم بودن دستورات باعث ساده شدن واحد کنترل ومسیرهای ارتباطی میشود که منجر به کم شدن تعداد ترانزیستورهای مورد نیاز برای ساخت پردازنده میشود اینکار سرعت پردازنده را نیز افزایش میدهد..
  - در ضمن میتوان واحد کنترل را بروش سیم بندی ساخت.

ویژگی های RISC

- دسترسی به حافظه محدود به دستورات load , Store است
  - از آنجائیکه عمل رجوع به حافظه معمولا در یک سیکل امکانپذیر نمیباشد،تمهیداتی در کامپایلر استفاده میشود تا pipelineبطور موثرتری استفاده شود.
  - باقی دستورات بر روی محتوی رجیسترها عمل میکنند.

ویژگی های RISC

- تعداد مد های آدرس دهی آنها کم است
  - معمولا فقط از مدهای آدرس دهی زیر استفاده میشود:
    - register addressing
    - direct addressing
    - register indirect addressing
    - displacement addressing

ویژگی های RISC

- دستورات طول و فرمت یکسانی دارند
- اینکار باعث میشود تا خواندن دستورات و رمزگشائی آنها سریع و ساده باشد.زیرا لازم نیست تا معلوم شدن طول دستور برای رمزگشایی آن صبر کرد.
- یکسانی فرمت باعث سهولت رمزگشایی میگردد زیرا کد و ادرس همه دستورات در محل یکسانی قرار دارند.

ویژگی های RISC

- تعداد رجیسترهای زیادی دارند.
  - کم شدن دستورات باعث کوچگ شدن واحد کنترل و آزاد شدن فضا برای گنجاندن تعداد بیشتری رجیستر میشود.
  - متغیرهای محلی، نتایج میانی و پارامترهای توابع در داخل رجیسترها ذخیره شده و تعداد رجوع به حافظه کاهش پیدا میکند.

سایر ویژگی های RISC

- استفاده ازتکنیک pipelining که باهمپوشانی سیکل های واکشی، رمزگشائی و اجرا شده و نهایتا به اجرای هر دستور در یک سیکل منجر میشود.
- تعداد زیادی رجیستر در داخل cpu وجود دارد
- استفاده از تکنیک همپوشانی register windows که باعث افزایش سرعت صدا زدن تابغ و بازگشت از آن میشود.
- حمایت از کامپایلر برای افزایش کارائی در ترجمه برنامه های سطح بالا

مقایسه RISC, CISC

- برای مثال ترجمه دستور a = a * b برای دو معماری فوق بصورت زیر خواهد بود:

CISC	MULT a, b
RISC	LOAD R1, a LOAD R2, b MUL R1,R2 STORE a, R1

- برای مثال ترجمه دستور a = a * b برای دو معماری فوق بصورت زیر خواهد بود:

- در معماری CISC سعی بر این است که با تعداد کمی دستورزبان ماشین عمل مورد نظر انجام شود .در حالیکه درRISC سعی بر این است که از دستوراتی استفاده شود که هر دستور در یک پالس قابل اجرا باشند .در نتیجه برنامه RISC دارای چندین خط دستور خواهد بود که احتیاج به حافظه بیشتری دارد.

مقایسهRISC, CISC

- فرمول محاسبه کارائی
- اگر از فرمول زیر برای محاسبه کارائی کامپیوتر استفاده شود:

$$\frac{\text{time}}{\text{program}} = \frac{\text{time}}{\text{cycle}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{instructions}}{\text{program}}$$

RISCسعی میکند تا این قسمت کاهش پیدا کند

CISCسعی میکند تا این قسمت کاهش پیدا کند

پیشگامان RISC

- ایده RISC در دهه 80 معرفی گردید .از جمله پیشگامان آن میتوان به موارد زیر اشاره نمود

  - IBM 801
  - Stanford MIPS
  - Berkeley RISC 1 and 2

کاربردهای RISC

- عمده پردازنده های کامپیوترهای سرور و workstarion از نوع RISC هستند.
- تعدادی از پردازنده های مشهور مبتنی بر RISC عبارتند از:
  - ARM/StrongARM که در تلفنهای همراه استفاده میشود
  - MIPS Rxx00 که در برخی دستگاه های بازی مورد استفاده است.
  - Hitachi SH مورد استفاده در سیستم های embedded
  - POWER/PowerPC مورد استفاده در کامپیوترهایMAC
  - Intel Pentium II and IIIاگرچه از معماری CISC پیروی میکنند ولی بسیاری از ویژگی های RISC را استفاده میکنند.

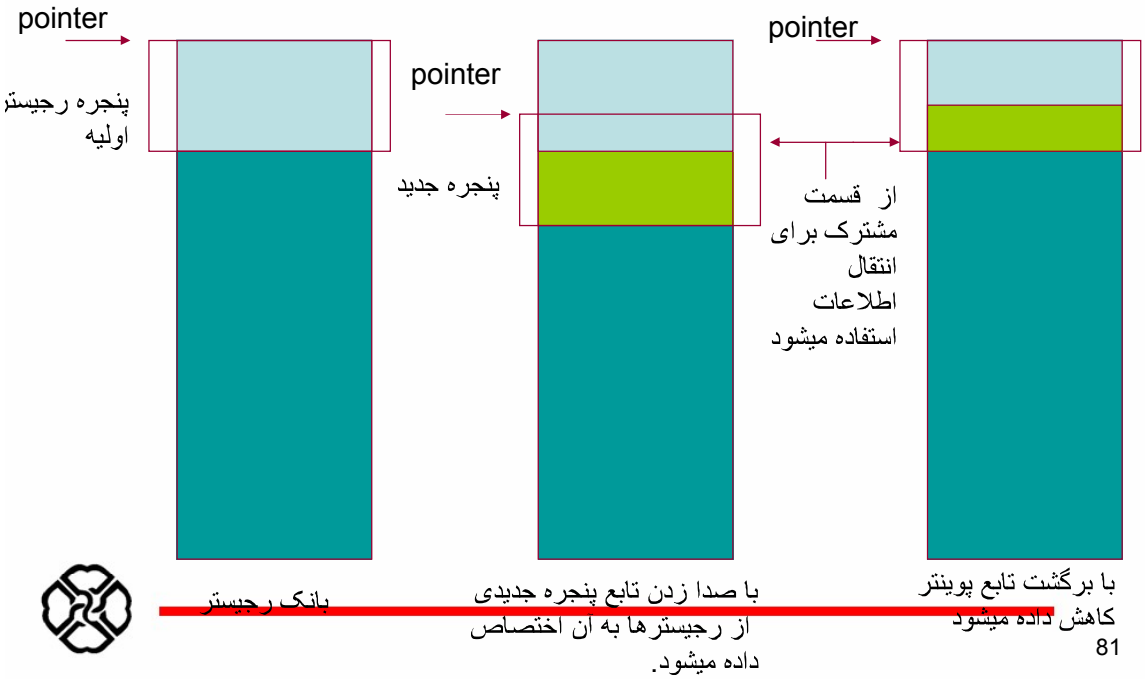
overlapped register windows

- دیدیم که صدا زدن برنامه فرعی و بازگشت از آن بخش مهمی از اجرای برنامه را بخود اختصاص میدهد .لذا سرعت بخشیدن به آن میتواند تاثیر زیادی در افزایش کارائی پردازنده داشته باشد.
- هنگام صدا زدن یک برنامه فرعی دستوراتی استفاده میشوند که مقادیر رجیسترها را ذخیره کنند، و پارامترها را انتقال دهند.
- برای بازگشت از برنامه فرعی مقدار بازگشتی تابع به برنامه صدا زننده منتقل شده و مقادیر ذخیره شده رجیسترها بازیابی میشوند.

overlapped register windows

- دربرخی از معمارهای RISC برای سهولت انتقال بین توابع از همپوشانی پنجره ای رجیسترها استفاده میشود تا نیاز به ذخیره و بازیافت مقادیر رجیسترها را حذف نمایند.
- هر زمان که تابعی که صدا زده میشود پنجره ای از رجیسترها از میان بانک رجیسترموجود در پردازنده به آن اختصاص میابد.
- اینکار از طریق افزایش پویتری انجام میشود .با برگشت از تابع پویتر به مقدار قبلی آن کاهش داده میشود.
- پنجره مورد استفاده برای تواب همجوار همپوشانی دارند تا امکان به اشتراک گذاشتن داده ها بین دو تابع وجود داشته باشد.

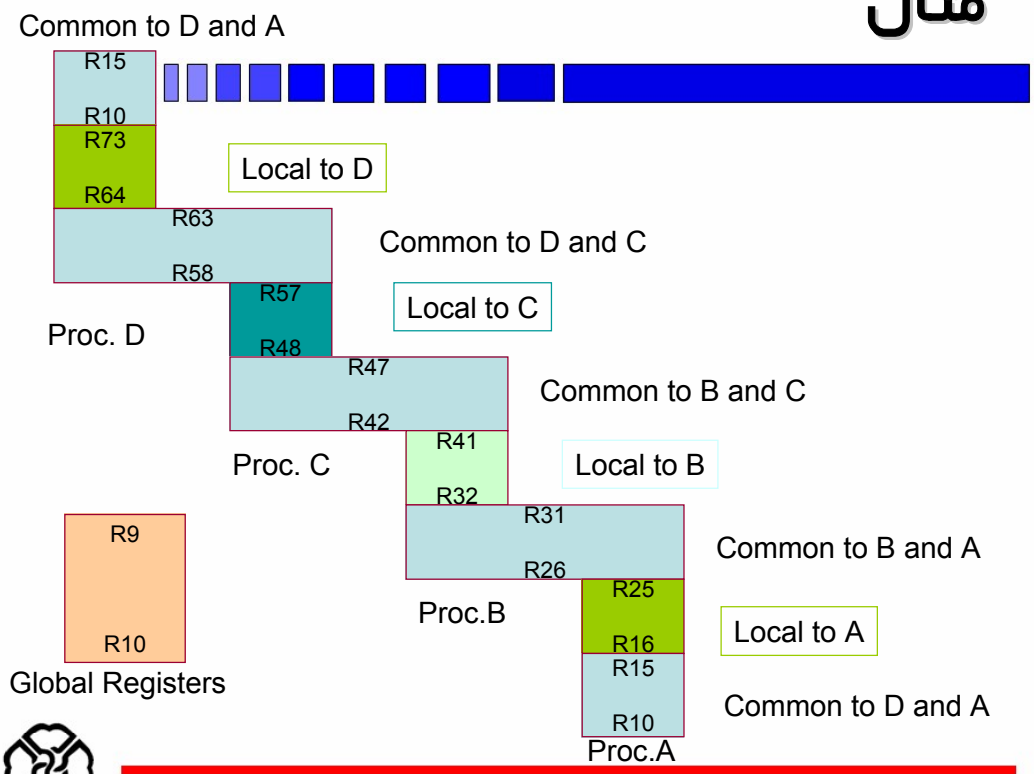
overlapped register windows



مثال

- در کتاب سیستمی معرفی شده است که دارای بانک رجیستری با مشخصات زیر است:
  - این بانک دارای 74 رجیستر است.
  - رجیستر های R0 تا R9بعنوان رجیسترهای عمومی قابل استفاده تمامی توابع هستند
  - بقیه رجیسترها به به 4 گروه A,B,C,Dتقسیم شده اند .
  - هر پنجره دارای 10 رجیستر اختصاصی و دو دسته رجیستر 6 تائی مشترک با با پنجره های مجاور است.
  - از رجیسترهای اختصاصی برا ذخیره متغیرهای محلی و از رجیسترهای مشترک برای انتقال پارامترها استفاده میشود.
  - به این ترنیب هر تابع میتواند از 32 رجیستر استفاده نماید

مثال



overlapped register windows

- نحوه محاسبه رجیسترهای قابل استفاده برای هر تابع به صورت زیر است.
  - G =تعداد رجیسترهای عمومی
  - L =تعداد رجیسترهای اختصاصی هر پنجره
  - C =تعداد رجیسترهای مشترک بین دو پنجره
  - W =تعداد پنجره ها

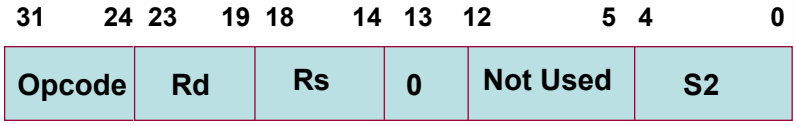
اندازه پنجره= L+2C+G

تعداد کل رجیسترها در بانک رجیستری= (L+C)W+G

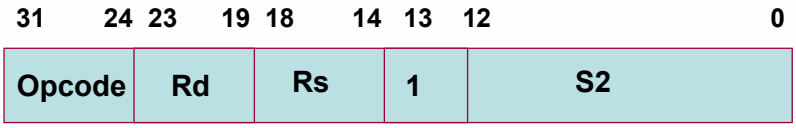
مثالی از پردازنده RISC: Berekely RISC I

- یکی از اولین معماری های RISC است که در دانشگاه برکلی مطرح شد.
- این پردازنده 32 بیتی:
  - دارای آدرس 32 بیتی بوده و میتواند از داده های8 و16و 32بیتی استفاده نماید.
  - سه مد آدرس دهی دارد :رجیستری، بلادرنگ و نسبی
  - یک بانک رجیستری 138 تائی شامل 10 رجیستر عمومی و 8 پنجره 32رجیستری دارد.
  - 31دستورالعمل دارد

فرمت دستورات Brekely RISC I



Register mode: S2=register



Register Immediate mode: S2=operand

PC relative Mode