دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

طراحی کامپیوتری سیستم‌های دیجیتال

تمرین اول

پاییز ۱۴۰۰

در این تمرین، به منظور یادآوری مفاهیم مربوط به طراحی مدارهای منطقی، از شما خواسته می‌شود یک کنترل‌کننده و مسیر داده برای مدار خواسته شده، **روی کاغذ** طراحی کنید. در تمرین بعدی، این طراحی را با استفاده از زبان توصیف سخت‌افزاری Verilog پیاده‌سازی خواهید کرد.

مقدمه

بازگشت (recursion)، روشی قدرتمند برای توصیف بسیاری از الگوریتم‌ها می‌باشد. خیلی از برنامه‌های نرم‌افزاری از این روش استفاده می‌کنند. به طور معمول در این برنامه‌ها در حین اجرا از یک پشته (stack) برای ذخیره‌سازی اطلاعات استفاده می‌شود. گاهی نیاز است که برای افزایش کارایی این برنامه‌ها، آن‌ها را به صورت یک مدار دیجیتالی پیاده‌سازی کنیم. در این تمرین، به منظور آشنایی با نحوه ایجاد و طراحی توابع بازگشتی در سخت‌افزار، از شما خواسته می‌شود تا مداری را طراحی کنید که یک مدل خاص از سری فیبوناچی را ایجاد می‌کند.

برای روشن‌تر شدن موضوع، به مثال زیر توجه کنید.

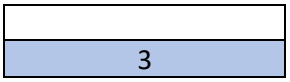
تابع زیر را که به زبان C پیاده‌سازی شده است، در نظر بگیرید:

```
int fib (int n)
{
    If (n <= 1)
        return 1;
    return fib (n – 1) + fib (n – 2);
}
```

در این تابع عنصر nام سری فیبوناچی به صورت بازگشتی محاسبه می‌شود. هربار که تابع fib صدا زده می‌شود، مقدار ورودی (یعنی n) و مقادیر بازگشتی تابع fib (برای مقادیر $n > 1$) در یک قاب پشته (stack frame) ذخیره می‌شوند. مقادیر بازگشتی، در ابتدا

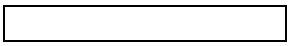
خالی و در مراحل بعدی مقدار می‌پذیرند. به طور ساده، وضعیت پشته در مراحل مختلف محاسبه‌ی مقدار fib(3) می‌تواند به شکل زیر باشد:

با صدا شدن تابع fib(3)، مقدار ۳ روی پشته قرار می‌گیرد.



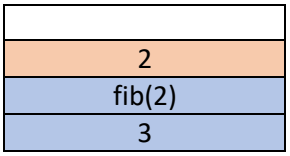
(2)

در ابتدا پشته خالی است.



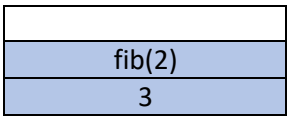
(1)

حال با صدا شدن fib(2)، قاب جدیدی به پشته اضافه می‌شود و مقدار ورودی در آن قرار می‌گیرد.



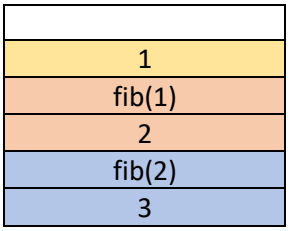
(4)

با توجه به اینکه 3 > 1، جای fib(2) روی قاب پشته درنظر گرفته می‌شود، ولی هنوز مقدار ندارد.



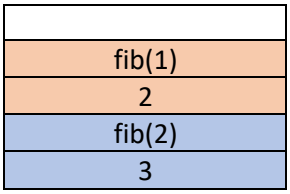
(3)

حال با صدا شدن fib(1)، قاب جدیدی به پشته اضافه می‌شود و مقدار ورودی در آن قرار می‌گیرد.



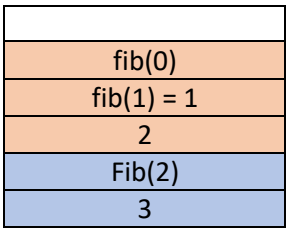
(6)

با توجه به اینکه 2 > 1، جای fib(1) روی قاب پشته درنظر گرفته می‌شود، ولی هنوز مقدار ندارد.



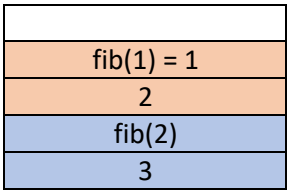
(5)

سپس جای fib(0) روی قاب پشته درنظر گرفته می‌شود، ولی هنوز مقدار ندارد.



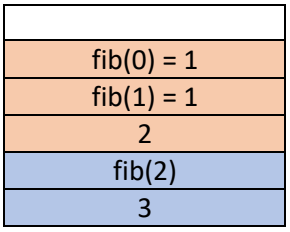
(8)

از آنجایی که مقدار 1 = 1 است، قاب بالای پشته پاک شده، بالاترین خانه‌ی پشته(fib(1))، با مقدار ۱ مقداردهی می‌شود.



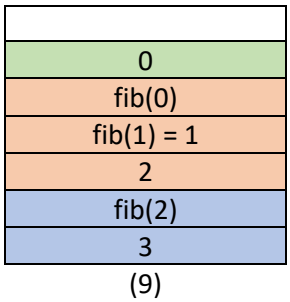
(7)

از آنجایی که مقدار 1 > 0 است، قاب بالای پشته پاک شده، بالاترین خانه‌ی پشته(fib(0))، با مقدار ۱ مقداردهی می‌شود.



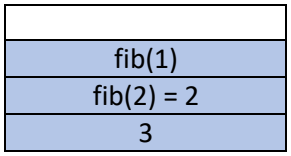
(10)

حال با صدا شدن fib(0)، قاب جدیدی به پشته اضافه می‌شود و مقدار ورودی در آن قرار می‌گیرد.



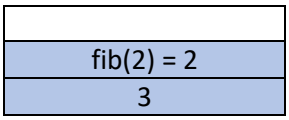
(9)

سپس جای fib(1) روی قاب پشته درنظر گرفته می‌شود، ولی هنوز مقدار ندارد.



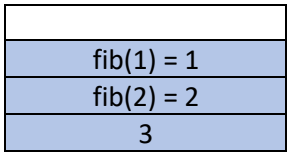
(12)

اکنون از آنجایی fib(2) قابل محاسبه است، قاب بالای پشته پاک شده، بالاترین خانه‌ی پشته(fib(2))، با مقدار محاسبه شده، مقداردهی می‌شود.



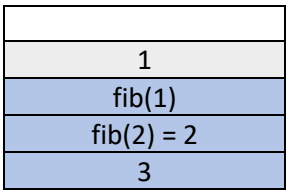
(11)

از آنجایی که مقدار 1 = 1 است، قاب بالای پشته پاک شده، بالاترین خانه‌ی پشته(fib(1))، با مقدار ۱ مقداردهی می‌شود.



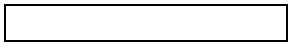
(14)

حال با صدا شدن fib(1)، قاب جدیدی به پشته اضافه می‌شود و مقدار ورودی در آن قرار می‌گیرد.



(13)

اکنون از آنجایی fib(3) قابل محاسبه است، قاب بالای پشته پاک شده، از آنجایی که پشته خالی شده، مقدار محاسبه شده باز گردانده می‌شود.


(15)

*هر قاب پشته، با یک رنگ جدا نمایش داده شده است.

توضیحات پروژه

تابع فیبوناچی زیر را در نظر بگیرید. (مقدار N مفروض است.)

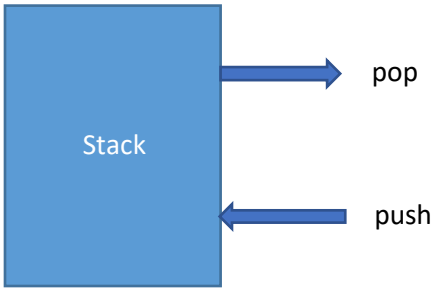
```
int fib (int n)
{
        If (n <= 1)
                return 1;
        else
        {
                If (n > N/2)
                        return ((n - 1) * fib (n - 1) + (n - 2) * fib (n - 2));
                else
                        return ((n - 2) * fib (n - 1) + (n - 1) * fib (n - 2));
        }
```

3

}

کنترل‌کننده و مسیر داده‌ی مداری را طراحی کنید که مقدار N را به عنوان ورودی دریافت کرده و عنصر Nام سری فیبوناچی را **طبق روال تابع بالا** (بصورت بازگشتی) محاسبه کند.

در طراحی مسیر داده، باید از یک پشته استفاده کنید.



یادآوری: پشته یک ساختمان داده برای نگه‌داری دنباله‌ای از اطلاعات است. رابط(interface) هر پشته، باید دو تابع push و pop را پشتیبانی کند. تابع push مقدار جدیدی را به بالای پشته اضافه می کند و تابع pop مقدار بالای پشته را (درصورت وجود) می خواند و حذف می‌کند. (در این پروژه به منظور کمتر شدن تعداد push و popها می‌توانید تابع top را نیز در رابط پشته در نظر بگیرید که مقدار بالای پشته را خوانده ولی حذف نمی‌کند.)