ساختار کامپیوتر و میکروپروسسور

دکتر باقری شورکی

مهندسی برق

برنا خدابنده ۴۰۰۱۰۹۸۹۸

پروژه ترم

طراحی یک میکروپروفسور

۱۰ تیر ۱۴۰۲

ساختار کامپیوتر و میکروپروسسور

پروژه ترم

برنا خدابنده ۴۰۰۱۰۹۸۹۸

تعریف پروژه

به دنبال طراحی دستگاهی هستیم که به طور خلاصه، شبیه ساز یک پردازنده، داخل یک پردازنده، است. چنین دستگاهی وظیفه دارد تا به صورت Real-time **برنامه نویسی** شود. و همینطور در اتمام برنامه، برنامه نوشته شده را اجرا نیز کند، بدین صورت که داخل پردازش خود، باید مراحلی که یک پردازنده طی میکند اعم از Fetch, Decode, Execute را داخل خود شبیه سازی کند، ولی این کار باید کاملا نرم افزاری باشد یعنی در عمل سخت افزار یک پردازنده را به صورت نرم افزاری باید شبیه سازی کنیم.

طراحی چنین ماشینی در عمل انجام شدنی است زیرا تمامی مراحل طی شده اعم از اجرا کد و نوشته شدن کد، یک مسئله تشخیص پذیر است و میتوان آن را به صورت یک Finite State Machine پیاده سازی کرد.

توصیف عملکرد

فرایند برنامه نویسی : این فرایند در واقع فقط نیاز دارد تا در آدرس های حافظه دلخواه، داده دلخواه توسط کاربر وارد و ذخیره شود، این اطلاعات میتوانند هم به صورت OPCODE و هم به صورت OPERAND باشند، برای پردازنده ما تفاوتی وجود ندارد.

روند برنامه نوشتن به شکل زیر است.

۱. دکمه ORG زده میشود.

۲. یک عدد ۲ رقمی به عنوان آدرس وارد میشود.

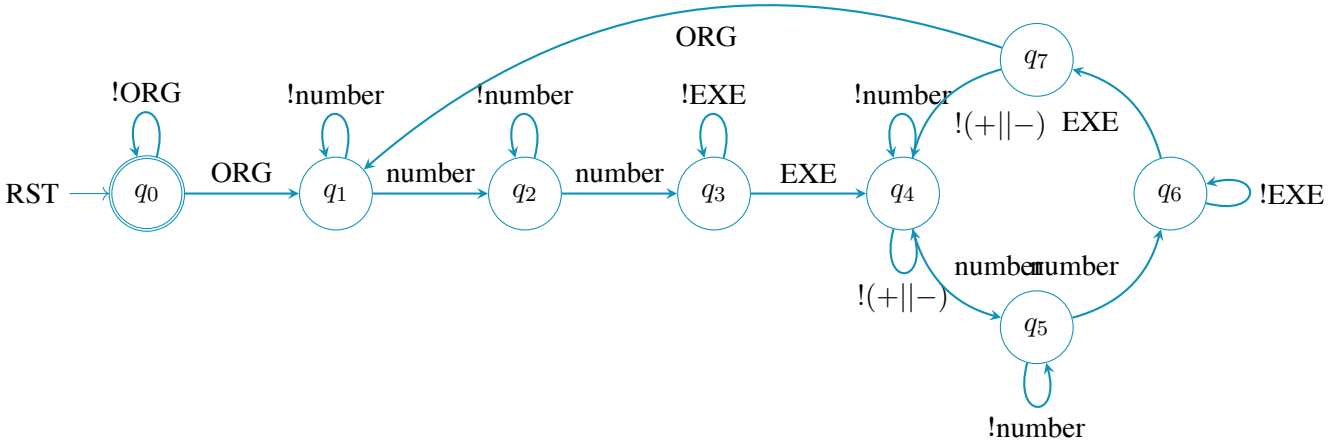
۳. دکمه EXE برای تایید زده میشود.

۴. در این آدرس عددی ۲ رقمی به عنوان OPCODE یا OPERAND قرار داده میشود.

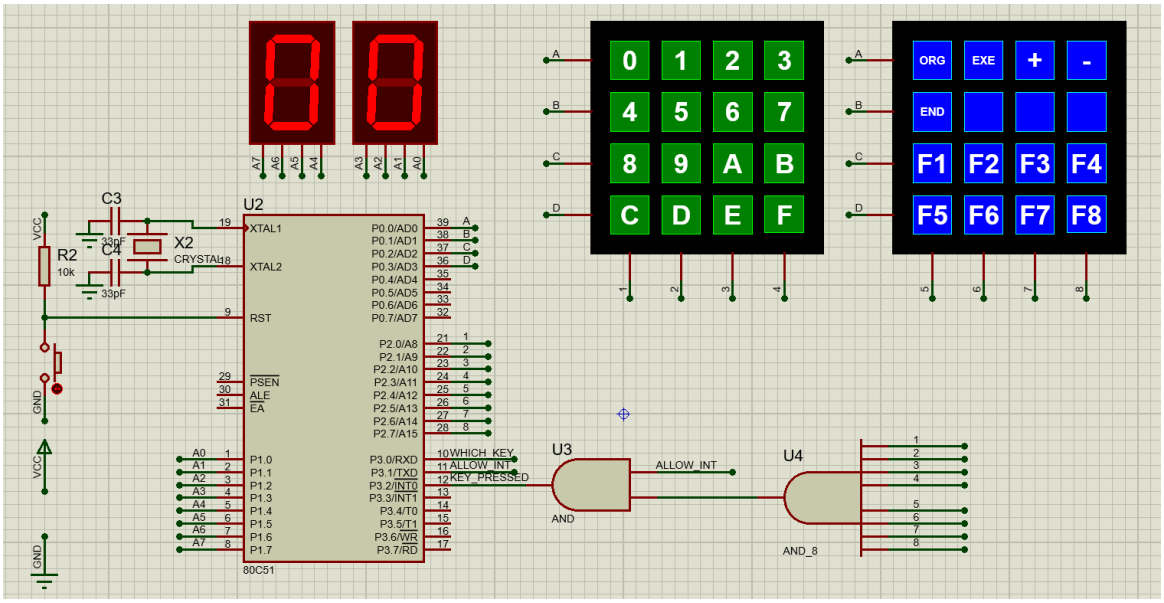
۵. دکمه EXE برای تایید زده میشود.

۶. با کلید های + یا - و یا دوباره با ORG به آدرس جدید میرویم.

همچنین بعضی از کلید ها مانند F1-F8 را تعبیه کردیم که کار های خاص را انجام میدهد، برای مثال F1 هر زمانی زده شود محتوای Accumilator را نمایش داده و F2 برای ما state در حال حاضر را نشان میدهد.



شکل ۱: برنامه نویسی



شکل ۲: شماتیک طرح

چالش ها

برای حالت های مختلف، رجیستر ها را به شکل زیر توصیف میکنیم تا مشکلات کمتری را بر بخوریم.
با تعریف یک رجیستر جداگانه برای PC مشکلاتی که ممکن بود برای پیاده سازی JMP داشته باشیم از بین می رود.
فقط برای STACK نیاز داشتیم تا مکانی از حافظه را اختصاص دهیم و یک SPTR تعریف کنیم، ولی برا این منظور نیاز داشتیم رجیستر R۰ که دسترسی به آن به صورت آدرس را داریم را نیز از بین ببریم، پس میکروپروفسور ما از قابلیت ها رفرنس آدرس دیگر بهره مند نمیبود، میتوانستیم با استفاده از برنامه های پیچیده تر و توصیف بخشی از حافظه به عنوان STACK این عمل را انجام دهیم، ولی برای سادگی در این پیاده سازی فقط یک رجیستر را به عنوان STACK در نظر گرفته ایم.
کد کامل برای این ماشین درج شده است.

جدول ۱: آدرس‌ها

حالت کدنویسی	آدرس‌ها	آدرس‌ها
در حالت کدنویسی	R0	وضعیت FSM
	R1	ورودی
	R2	داده ورودی
	R3	آدرس ورودی
در حالت اجرا	R6	آدرس شروع
	R1	PC (شمارنده برنامه)
	R3	پشته
	R4	IR (ثبت دستور)

منبع: ایجاد توسط نویسنده.

حالت کد نویسی توصیف شده، حالت اجرا نیز قابل ساخت است، بدین صورت که کافسیست Execute decode، fetch، را شبیه سازی کنیم، در کد درج شده هست، ولی برای مثال برای پیاده سازی یک اینستراکشن خاص:

```
ADD_R2_MNEMONIC: CJNE R4, #02AH, ADD_R5_MNEMONIC
         ADD A, R2
     RET
```

تست کاربر

با استفاده از کد زیر، کارگفته شده را انجام میدهیم.

```
MOV R2, #n   ; Set n to the desired value
MOV R0, #m   ; Set R1 to the starting address m
MOV R5, #1   ; Initialize R2 to hold the current number
LOOP:
MOV A, R5
MOV @R0, A   ; Write the current number to the memory location pointed by R1

INC R0       ; Increment the memory address
INC R5       ; Increment the current number

; Decrement n and repeat the loop if not zero
DEC R2
MOV A, R2
JNZ LOOP
```

حال فقط کافیست تا که کد بالا را روی میکروپروفسور خود قرار دهیم. کد معادل به صورت عددی به شکل زیر است.

```
64: 7A (MOV R2, #data)
65: n
66: 78 (MOV R0, #data)
67: m
68: 7D (MOV A, #data)
69: 01
6A: ED (MOV A, R5)
6B: F6 (MOV @R0, A)
6C: 08 (INC R0)
6D: 0D (INC R5)
6E: 1A (DEC R2)
6F: EA (MOV A, R2)
70: 70 (JNZ reladdr)
71: 6A
```

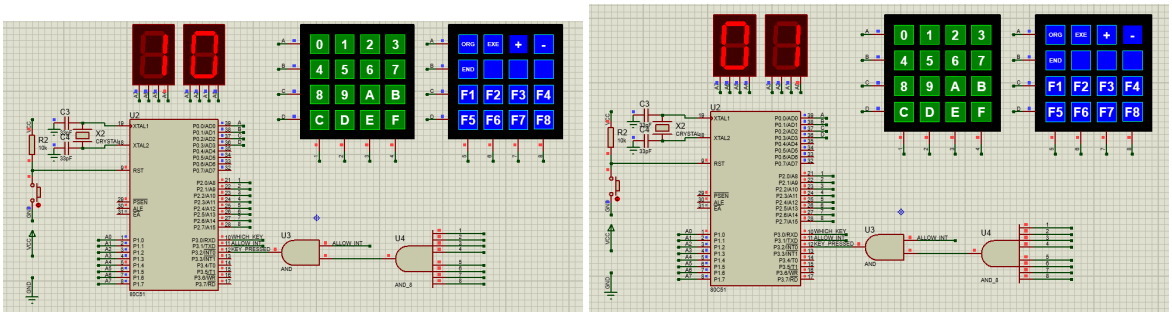
حال بعد از وارد کردن این کد داخل میکروپروفسور به طور توصیف شده و اجرا، میبینیم که این اتفاق انجام میشود.
بدین صورت که کد را قرار میدهیم، END را میزنیم، و سپس بعد از RESET با استفاده از ORG و یا +، - به آدرس مد نظر میرویم و میبینیم که داده مورد نظر در آنجا ذخیره شده استو.

چکیده

در آخر ما موفق به ساخت چنین دستگاهی شدیم، بدین صورت که حال میکروپروفسور ای داریم که هر کد اسمبلی ای که به آن دهیم را میتواند مانند یک پروسسور واقعی (با یک سری محدودیت بیشتر) به صورت Real-time اجرا کند و برنامه ریزی شود.
چالش های زیادی در این پیاده سازی داشتیم، اعم از کد کردن ورودی از صفحه کلید به اپراتور های داخل میکروپروفسور، ذخیره درست اطلاعات، پیاده سازی برنامه ریزی میکروپروفسور به صورت نرم افزاری و با صفحه کلید، توصیف برنامه نویسی به صورت یک FSM، پیاده سازی و آپدیت مناسب این FSM، و در نهایت چالش کمتری ولی چالش برای شبیه سازی سیکل اجرای برنامه در داخل پروسسور.
این قسمت آخر در واقع محدودیت اصلی ما را ایجاد کرده زیرا بعضی از رجیستر ها صرف رجیستر های داخلی این پروسسور نرم افزاری میشوند، همچنین این پروسسور نرم افزاری دیگر قابلیت تعامل با دنیای خارجی به صورت I/O را ندارد.
در نهایت با استفاده از حداقل رجیستر های پروسسور، و پیاده سازی مناسب FSM توانستیم سیکل های برنامه نویسی و سیکل اجرا را به صورت نرم افزاری شبیه سازی کنیم و یک میکروپروفسور بسازیم.

نتایج تست

تست شده با $m = A0$، $n = 10$
عکس زیر داده های خانه $A0$ و $B0$ است.

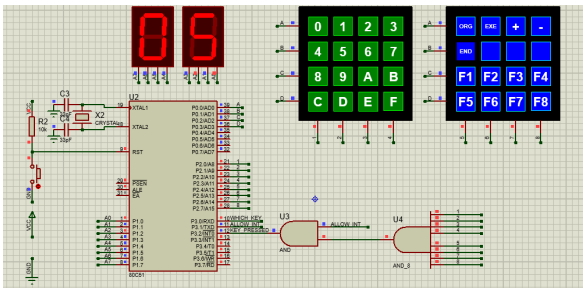


شکل ۳: اعداد اول و آخر ذخیره شده

تست های دیگر

کد برای محاسبه $m + n$:

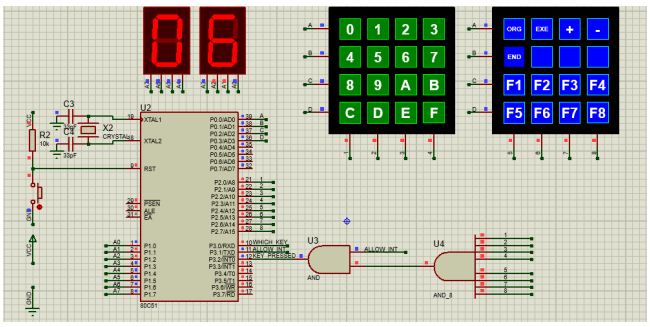
```
MOV A, #02H : 74, m
ADD A, #03H : 24, n
NOP         : 00
```



شکل ۴: نتیجه برای محاسبه ۳+۲

محاسبه برای $m * n$

```
MOV R4, n
MOV R7, m
MOV A, #00H ; result register
LOOP:
MOV A, R5
ADD A, R4
MOV R5, A
DEC R7
MOV A, R3
JNZ LOOP
```



شکل ۵: نتیجه برای محاسبه ۳*۲