باسمه تعالی

توضیحی کوتاه:

کد ها شامل چند بخش می شود. کد های مربوط به دیتاپت و کنترل یونیت و مموری.
تو وقتی که یک cpu درست می کنی برای امن که بتونی یک دستور رو اجرا کنی باید از یک سری ریز دستور استفاده کنی.با یک مثال توضیح می دهم. فرض کن می خواهی مقدار ثبات pc را یک واحد اضافه کنی و این کار در یک کلاک انجام شود. برای این کار باید سیگنال های کنترل یونیت را ابتدا تنظیم کنی که به این دستور خاص قبلا به صورت پیش فرض پیاده سازی شده که می تونی به صورت زیر فراخوانی کنی cpu.getCu.pc_pc_1_add.
بعد از این که سیگنال های کنترل یونیت را پیاده سازی کردی نوبت به آن می رسد تا با فراخوانی تابع next_clock تغییراتی که قصد انجام آن را داشتی را در دیتاپت و کنترل یونیت به انجام برسانی.
چند دستور برای انتقال بین رجیستر ها به عنوان مثال:

```
Pc = pc + 1:
        cpu.getCu.pc_pc_1_add();
        cpu.next_clock();
H = mbr:
        cpu.getCu.h_mbr();
        cpu.next_clock();
Pc = pc + 1, fetch
        cpu.getCu.pc_pc_1_add();
        cpu.getCu.fetch();
        cpu.next_clock();
```

برای راحتی شما دوست عزیز تمام زیر دستور های مورد نیاز به صورت پیش فرض ارایه شده که می تونی اون رو فراخوانی کنی.جدول زیر مربوط به این زیر دستورات می باشد.

Pc = pc + 1	pc_pc_1_add
Opc = mbr	opc_mbr
Mdr = mbr	mdr_mbr
H = 4	h_4
Mar = sp = sp + h	mar_sp_sp_h_add
Mar = sp = sp - h	mar_sp_sp_h_sub
H = mbr	h_mbr
Pc = pc + h	pc_pc_h_add
H = mbr + h	h_mbr_h_add

Mar = sp	mar_sp
H = mdr	h_mdr
Pc = pc - 1	pc_pc_1_sub
Mdr = mdr + h	mdr_mdr_h_add
Mdr = mdr - h	mdr_mdr_h_sub
Tos = mdr	tos_mdr
H = tos	h_tos
Mar = lv + h	mar_lv_h_add
Mar = cpp + H	mar_cpp_h_add
nop	nop

fetch	fetch
read	read
write	write
fetch_w	fetch_w
read_w	read_w

نکته: دستورات زیر به این صورت هست«

Fetch: این دستور آدرس را از ثبات mbr می گیرد و داده را از مموری می خواند و در باس(bus) خروجی از مموری قرار می گیرد.

Fetech_w: این دستور داده را از باس خروجی از مموری می گیرد و در ثبات mbr ذخیره می کند

Read: این دستور آدرس را از ثبات mdr می گیرد و داده را از مموری می خواند در باس خروحی از مموری قرار می گیرد.

Read_w: این دستور داده را از باس خروجی از مموری می گیرد و در ثبات mdr ذخیره می کند.

Write: این دستور داده را از ثبات mdr می گیرد و در آدرسی که در ثبات mdr ذخیره شده است در مموری ذخیره می کند.

حال یه شرح یک مثال می پردازیم و به طور مثال دستور BIPPUSH را طبقRTL زیر پیاده سازی می کنیم

RTL:
PC = PC + 1, FETCH
FETCH_W
OPC = MBR
PC = PC + 1, FETCH
FETCH_W
MDR = MBR
H = 4
MAR = SP = SP + H
WRITE

کد دستور های بالا در جاوا:

```
CPU cpu = new CPU();
cpu.reset();
```

//کد بالا مربوط به ساخت سی پی یو هست.

```
cpu.getCu().pc_pc_1_add();
cpu.getCu().fetch();
cpu.next_clock();
cpu.getCu().nop();
cpu.getCu().fetch_w();
cpu.next_clock();
cpu.getCu().opc_mbr();
cpu.next_clock();
cpu.getCu().pc_pc_1_add();
cpu.getCu().fetch();
cpu.next_clock();
cpu.getCu().pc_pc_1_add();
cpu.getCu().fetch();
cpu.next_clock();
cpu.getCu().nop();
cpu.getCu().fetch_w();
cpu.next_clock();
cpu.getCu().opc_mbr();
cpu.next_clock();
cpu.getCu().pc_pc_1_add();
cpu.getCu().fetch();
cpu.next_clock();
```

حال به توضیح نحوره کار cpu می پردازیم.

دستورات زیر برای تمام دستورات در تمرین ۳ مشترک هستش

PC = PC + 1, FETCH
FETCH_W
OPC = MBR
PC = PC + 1, FETCH

کار این دستور این است که آپکد مربوط به دستور را از مموری دریافت می کند و به ثبات opc انتقال می دهد که تا آخر اجرای کامل دستورات مربوط به این آپکد ثبات opc ثابت می ماند.با توجه به این ثبات می توان تصمیم گرفت چه سلسله از دستوراتی را در ادامه انجام داد.