گزارش پروژه AUT-MIPS

پروژه پایانی درس آزمایشگاه معماری کامپیوتر

استاد

سرکار خانم کامران

اعضای گروه

پویا پارسا (۹۲۳۱۰۰۵)

جواد هاشمی (۹۰۳۱۰۵۲)

نگاه کلی

مقدمه

هدف از انجام این پروژه پیاده سازی و طراحی یک پردازنده ی شبه MIPS به نام AUT-MIPS بود.

طول داده های این پردازنده ۱۶ بیتی هستند و PC هر مرحله با 2 جمع می شود.

برای توسعه این پروژه از محیط Zamia Cad و زبان پیاده سازی VHDL استفاده شده.

پیاده سازی Register File

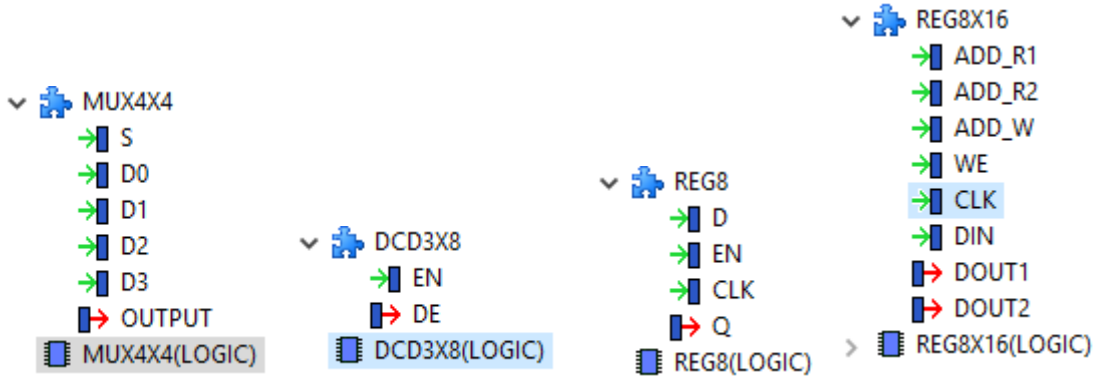
گام اول ایجاد یک Register File ، پیاده سازی یک کامپوننت رجیستر ۸ بیتی به نام REG8 انجام شد. کامپوننت های بعدی به ترتیب MUX4x4 و DCD3x8 هستند. سپس کامپوننت اصلی registry file (REG8X16) پیاده سازی شد.

رجیستر فایل شامل ۸ خط داده ی ۱۶ بیتی است که هر خط از دو REG8 تشکیل شده.

دو آدرس ورودی برای Read و یک آدرس برای مشخص کردن مقصد Write تعریف شده اند.

جهت عملیات نوشتن باید ورودی های WE(1) و ADD_W(3) مقدار دهی شده باشند.

خروجی این کامپوننت دو خروجی DOUT1 و DOUT2 می باشد.



پیاده سازی مسیر داده (Data Path)

برای پیاده سازی دستورات R-Type نیاز به یک واحد محاسبه ALU ۱۶ بیتی و مسیر داده داریم.

در مرحله ی اول، کامپوننت های پایه یعنی ADD4 – BWAND4 و BWOR4 پیاده سازی شدند.

برای پیاده سازی تفرق کننده، یک کاپمونت کمکی برای مکمل ۱ به نام PINV4 استفاده شد.

واحد ALU4 توسط کامپوننت های کمکی پیاده سازی شد. و در نهایت با استفاده از ۴ واحد ALU4 واحد محاسباتی اصلی ALU16 پیاده سازی شد.

پیاده سازی دستورات I-Type و حافظه برای دستور عمل ها

ابتدا دو کامپوننت مرتبط با عملیات دریافت دستورالعمل و افزایش PC (INCTWO) پیاده سازی شدند. برای INCTWO نیاز به ADD16 شد.

برای نگه داری دستور عمل ها، از یک کامپوننت که دستور عمل ها در آن به طور دستی وارد شده اند استفاده می کنیم.(INS_MEM)

پیاده سازی حافظه داده ها Data Memory

مرحله ی بعدی پیاده سازی حافظه برای نگهداری داده ها (DAT_MEM) است. این حافظه از ۵۱۲ خط ۱۶ بیتی تشکیل شده و خانه های ۲۵۶ و ۲۵۸ به طور نمونه با داده های ۲۵۶ و ۲۵۸ (برابر آدرسشون ☺) مقدار دهی اولیه شده اند. (بقیه ۰) با استفاده از ورودی های RE و WE می توان عملیات خواندن یا نوشتن را انجام داد.

پیاده سازی مسیر کنترل Control Path و دستورات Branch

برای پیاده سازی این بخش از کامپوننت های SHLONE (برای شیف دادن به سمت چپ عدد ۱۶ بیتی) و MUX3X16استفاده شده.همچنین نیاز به پیاده سازی کامپوننت های کمکی MUX2X16 و MUX2X3 بود.

برای عملیات پرش Jump یک کامپوننت به خصوص به نام PCJMP ساخته شد که ۳ بیت پر ارزش PC و ۱۲ بیت آدرس پرش را (با یک صفر در قسمت کم ارزش) ادغام می کند.

اتصال نهایی و تکمیل مسیر داده

برای تکمیل مسیر داده، دو کامپوننت PC و ماژول Control نیاز بود.

برای پیاده سازی PC از سه سیگنال CLK (Clock), En (Enable) و RST برای مقدار دهی ۰ استفاده شد.

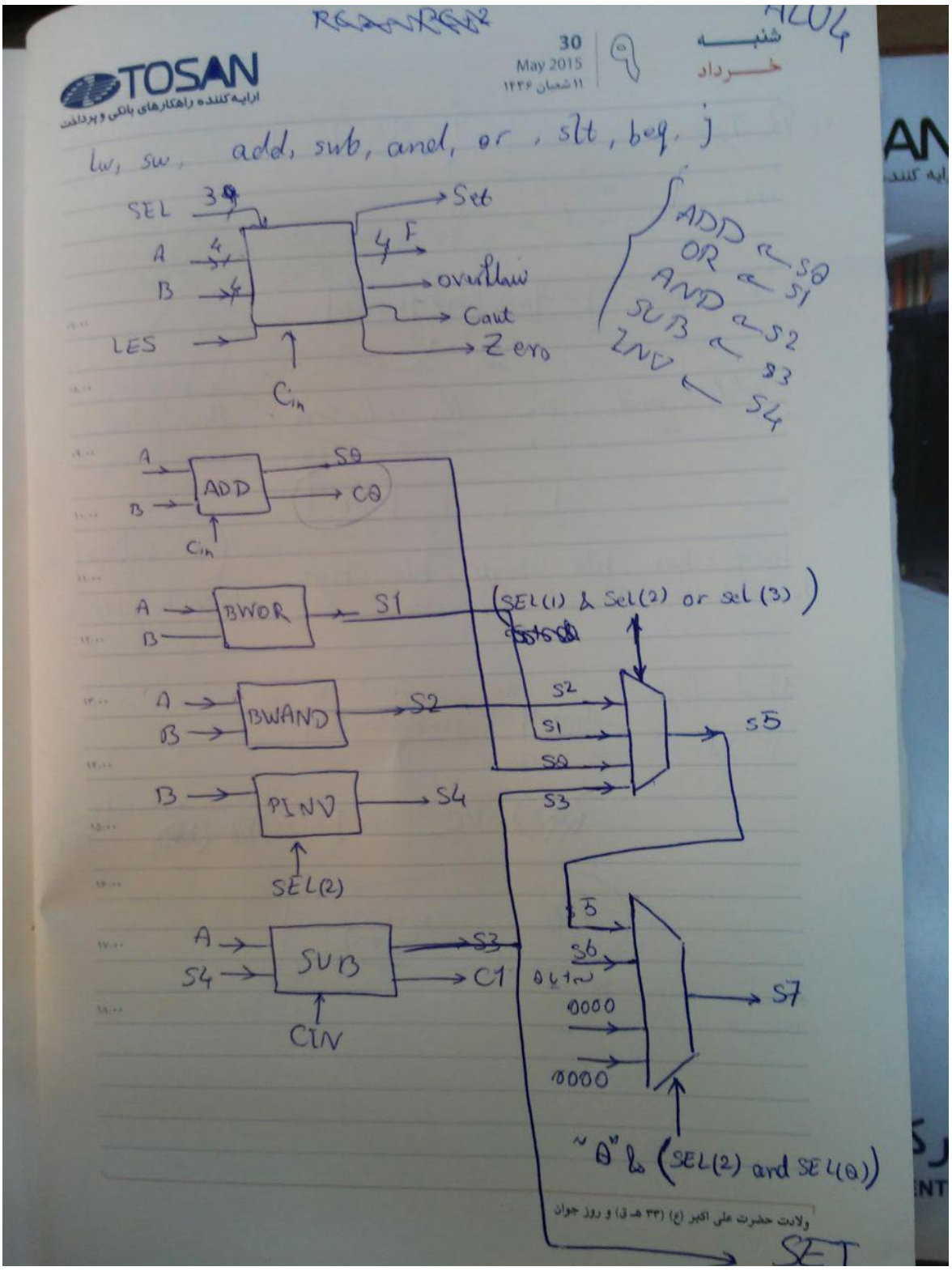
کامپوننت CONTROL برای تشخیص OPCode و تولید سیگنال های مورد نیاز مورد استفاده قرار می گیرد.

لیست سیگنال های کنترلی

عملیات	سیگنال
نوشتن در رجیستر فایل	RegDST
هنگام پرش	Branch
خواندن از حافظه	MemRead
خواندن موقعیت حافظه از رجیستر	MemToReg
کد دستور العمل	ALUOp
هنگام نوشتن در حافظه	MemWrite
انتخاب منبع واحد محاسباتی	ALUSrc
نوشتن در یک رجیستر	RegWrite

پایان :)

جزئیات پیاده سازی همراه با توضیحات کامل در سورس کد پروژه قابل بررسی هستند.


TOSAN
ارایه کننده راهکارهای بانکی و پردازشی
30 May 2015
۱۱ شعبان ۱۴۳۶
شنبه
خرداد
lw, sw, add, sub, and, or, slt, beq, j
SEL
A
B
LES
Cin
Set
overflow
Cout
Zero
ADD ← S0
OR ← S1
AND ← S2
SUB ← S3
INV ← S4
ADD
CO
BWOR
S1
BWAND
S2
PINV
S4
SEL(2)
SUB
S3
C1
CIN
(SEL(1) & Sel(2) or sel(3))
S5
S6
0000
S7
~"0" & (SEL(2) and SEL(0))
SET