به نام خدا

SAYEH Basic Computer

سید نوید کرمی نژاد

سید سینا ملکوتی

Amirkabir
University of Technology

۱. هدف

طراحی و پیاده سازی یک پردازنده به نام سایه

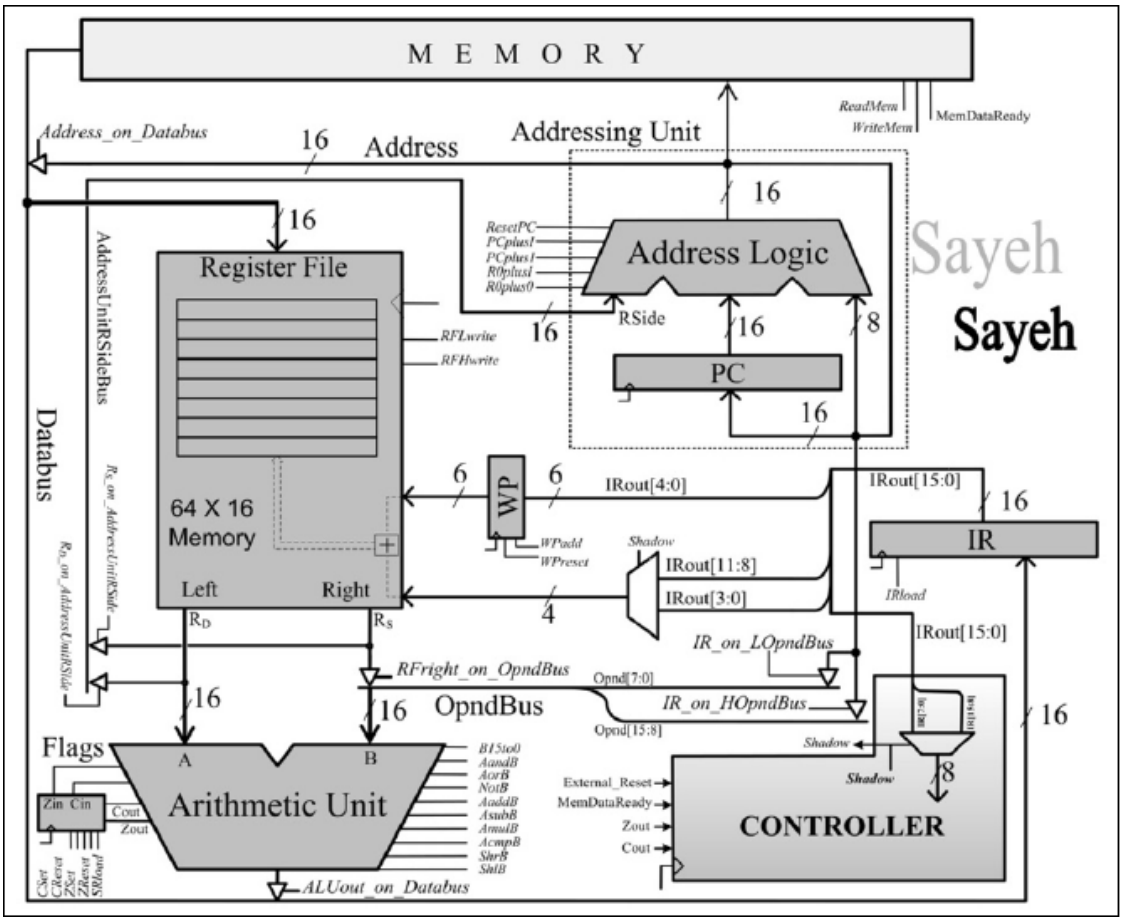
(SAYEH : Simple Architecture, Yet Enough Hardware) که شامل

دو بخش زیر است :

- کنترلر (Controller)
- مسیر دیتا (Data Path)

سایه با یک حافظه در ارتباط است. نحوه اتصال اجزا به هم به صورت شکل زیر

است:



۲. حافظه (Memory)

حافظه مورد استفاده یک حافظه 16*1024 است.

ورودی ها :

- Clk
- ReadMem : زمانی که این ورودی ۱ باشد داده در حافظه ذخیره می شود.
- WriteMem : زمانی که این ورودی ۱ باشد داده از حافظه خوانده می شود.
- AddressBus : این ورودی شماره خانه ای از حافظه که قرار است روی آن نوشته شود یا از آن خوانده شود، مشخص می کند.
- DataBus : این سیگنال یک سیگنال ورودی – خروجی ۱۶ بیتی است.

خروجی ها :

- MemDataReady : این یک سیگنال کنترلی است که وقتی ۱ باشد می توان روی حافظه نوشت یا از آن خواند.

- DataBus : این سیگنال یک سیگنال ورودی – خروجی ۱۶ بیتی است.

۳. سایه

هر کدام از اجزای سایه در بخش های جداگانه ای بررسی می شود.

۳-۱ : DataPath

مسیر داده به طور کلی، تعدادی ورودی و خروجی دارد که ورودی های آن همه از Controller و خروجی های آن از اجزای درونی خودش می آید. اجزای تشکیل دهنده DataPath عبارت اند از:

- Addressing Unit: این ماژول خروجی به ما می دهد که مشخص می کند قرار است از کدام خانه حافظه خوانده و یا روی آن نوشته شود.
- Instruction Register: یک رجیستر ۱۶ بیتی است که ورودی دستور را از حافظه می گیرد و در اختیار کنترلر و Addressing Unit قرار می دهد.
- Register File: این ماژول شامل ۶۴ رجیستر ۱۶ بیتی است که داده های خروجی از حافظه، Arithmetic Unit و Addressing Unit را در خود

ذخیره می کند. نکته مهم این است که به طور همزمان تنها می توان به ۴ رجستر از ۶۴ رجیستر موجود دسترسی داشت.

- Window Pointer: با توجه به نکته بالا این ورودی ضروری به نظر می رسد که به یک رجیستر ۶ بیتی نیاز است که در آن شماره اولین خانه ای از Register File که به آن دسترسی داریم، ذخیره شود.
- Flag Register: این رجیستر دو مقدار Carry Flag و Zero Flag که به عنوان خروجی های Arithmetic Unit هستند برای استفاده دوباره در آن نگهداری کند.
- Arithmetic Unit: واحد محاسباتی که مسئول انجام دستورات زیر را انجام می دهد:

✓ A+B

✓ A*B

✓ A-B

✓ A and B

✓ A or B

✓ A xor B

✓ B 15 to 0

✓ Not B

✓ 2'complement B

- ✓ A compare B
- ✓ Shift Left B
- ✓ Shift Right B

ورودی ها:

- Clk
- Cset, Creset, Zset, SRLoad: با توجه به این که هر کدام از این بیت ها ۱ باشد، مقدار مربوطه از Flag Register خارج می شود.
- funcSelect: این سیگنال ۴ بیتی تعیین می کند که کدام عملیات محاسباتی در Arithmetic Unit انجام خواهد شد.
- AluOut_on_Databus: در صورت ۱ بودن این ورودی نتیجه عملیات محاسباتی روی سیگنال DataBus که در شکل نیز دیده می شود، قرار می گیرد.
- IRload: اگر این بیت کنترلی ۱ باشد ورودی IR مستقیما روی خروجی قرار می گیرد.
- WPadd: اگر این ورودی ۱ باشد، مقدار ورودی با مقدار ذخیره شده جمع و به خروجی منتقل می شود.
- WPreset: در صورت ۱ بودن مقدار خروجی ریست می شود.

- RFLwrite,RFHwrite: این بیت ها مشخص می کنند که قرار است ورودی Register File روی کدام بیت های رجیستر ذخیره شود. اگر RFL ۱ باشد ۸ بیت کم ارزش ورودی روی ۸ بیت کم ارزش رجیستر و اگر RFH ۱ باشه ۸ بیت کم ارزش روی ۸ بیت پر ارزش رجیستر ذخیره می شود. در صورتی که هردو همزمان ۱ باشند ورودی بطور کامل روی رجستر موردنظر نوشته می شود.
- EnablePC: اگر این بیت ۱ باشد مقدار ورودی PC مستقیما به خروجی وصل می شود.
- ResetPC: در صورت ۱ بودن این بیت مقدار PC ریست می شود.
- PCplusI: در صورت ۱ بودن این بیت مقدار PC با مقدار Immidate که در واقع ۸ بیت کم ارزش خروجی IR است جمع می شود.
- PCplus1: در صورت ۱ بودن این بیت مقدار PC با ۱ جمع می شود.
- RplusI: در صورت ۱ بودن این بیت مقدار Rside که ورودی اختصاصی AU است با مقدار Immidate جمع می شود.
- Rplus0: در صورت ۱ مقدار Rside مستقیما به خروجی AU منتقل می شود.

- Rs_on_AddressUnitRSide, Rd_on_AddressUnitRSide:
  اگر هر کدام از این بیت ها ۱ باشد(همزمان این دو بیت ۱ نمی شوند) مقدار Rs یا Rd که خروجی RF هستند به عنوان Rside وارد AU می شوند.
- Address_on_DataBus: این یک بیت کنترلی است که در صورت فعال بودن مقدار AU را روی DataBus قرار می دهد.
- inputFromMemory: این سیگنال ۱۶ بیتی مقداری است که از حافظه روی DataBus قرار می گیرد.
- ReadMem, WriteMem: به ترتیب اگر هر کدام از این بیت ها ۱ باشند داده روی حافظه نوشته و یا از آن خوانده می شود.

خروجی ها:

- outputToMemory: این مقداری است که به حافظه به عنوان ورودی داده می شود. این سیگنال ۱۶ بیت دارد.
- Cout: بیت خروجی اختصاصی Flag Register است که به عنوان ورودی به Controller داده می شود.
- Zout: بیت خروجی اختصاصی Flag Register است که به عنوان ورودی به Controller داده می شود.

- addressUnitOutputToMemory: عملکرد این سیگنال در قسمت "حافظه" توضیح داده شد.
- IR_output_to_controller: خروجی اختصاصی IR که برای Decode شدن به Controller منتقل می شود.

۲-۳ : Controller

این ماژول وظیفه فعال کردن بیت های کنترلی را دارد.

کنترلر، دستورات را از IR می گیرد و طبق جدول صفحه بعد بیت های کنترلی خاصی را فعال می کند.

در ضمن ورودی External Reset کل دستگاه را به حالت اولیه بر می گرداند.

Zout و Cout نیز از DataPath وارد Controller می شود.

Instruction Mnemonic and Definition		Bits 15:0	RTL notation: *comments or condition*
nop	No operation	0000-00-00	No operation
hlt	Halt	0000-00-01	Halt, fetching stops
szf	Set zero flag	0000-00-10	Z <= '1'
czf	Clr zero flag	0000-00-11	Z <= '0'
scf	Set carry flag	0000-01-00	C <= '1'
ccf	Clr carry flag	0000-01-01	C <= '0'
cwp	Clr Window pointer	0000-01-10	WP <= "000"
mvr	Move Register	0001-D-S	R_D <= R_S
lda	Load Addressed	0010-D-S	R_D <= (R_S)
sta	Store Addressed	0011-D-S	(R_D) <= R_S
inp	Input from port	0100-D-S	In from Rs write to R_D
oup	Output to port	0101-D-S	Out to port R_D from R_S
and	AND Registers	0110-D-S	R_D <= R_D & Rs
orr	OR Registers	0111-D-S	R_D <= R_D | Rs
not	NOT Register	1000-D-S	R_D <= ~R_S
shl	Shift Left	1001-D-S	R_D <= sla Rs
shr	Shift Right	1010-D-S	R_D <= sra Rs
add	Add Registers	1011-D-S	R_D <= R_D + Rs + C
sub	Subtract Registers	1100-D-S	R_D <= R_D - Rs – C
mul	Multiply Registers	1101-D-S	R_D <= R_D * Rs :*8-bit multiplication*
cmp	Compare	1110-D-S	R_D, Rs *(if equal:Z=1; if R_D<Rs: C=1)*
mil	Move Immd Low	1111-D-00-I	R_{DL} <= {8'bZ, I}
mih	Move Immd High	1111-D-01-I	R_{DH} <= {I, 8'bZ }
spc	Save PC	1111-D-10-I	R_D <= PC + I
jpa	Jump Addressed	1111-D-11-I	PC <= R_D + I
jpr	Jump Relative	0000-01-11-I	PC <= PC + I
brz	Branch if Zero	0000-10-00-I	PC <= PC + I :*if Z is 1*
brc	Branch if Carry	0000-10-01-I	PC <= PC + I :*if C is 1*
awp	Add Win pntr	0000-10-10-I	WP <= WP + I

ماشین حالت (State Machine) زیر شِمای کلی این جزء را نشان می دهد: