ریزپردازنده

کامپیوتری که هم اکنون به کمک آن در حال مشاهده و مطالعه این صفحه هستید، دارای یک ریزپردازنده است. ریزپردازنده به منزله مغز کامپیوتر است و تمامی کامپیوترها اعم از کامپیوترهای شخصی، کامپیوترهای دستی و ... دارای ریزپردازنده می باشند. نوع ریزپردازنده استفاده شده در یک کامپیوتر می تواند متفاوت باشد ولی تمامی آنان عملیات مشابهی را انجام می دهند.

تاریخچه ریزپردازنده ها

ریزپردازنده که CPU **هم نامیده می گردد، پتانسیل های اساسی برای انجام محاسبات و عملیات مورد نظر در یک کامپیوتر را فراهم می نماید. ریزپردازنده از لحاظ فیزیکی یک تراشه است. اولین ریزپردازنده در سال** 1971 **و با نام** Intel 4004 **معرفی گردید. ریزپردازنده فوق چندان قدرتمند نبود و صرفا" قادر به انجام عملیات جمع و تفریق چهار بیتی بود. نکته مثبت پردازنده فوق، استفاده از صرفا" یک تراشه بود. قبل از آن مهندسین و طراحان کامپیوتر از چندین تراشه و یا عصر برای تولید کامپیوتر استفاده می کردند.**

اولین ریزپردازنده ای که بر روی یک کامپیوتر خانگی نصب گردید، 8080 **بود. پردازنده فوق هشت بیتی و بر روی یک تراشه قرار داشت. این ریزپردازنده در سال** 1974 **به بازار عرضه گردید. اولین پردازنده ای که باعث تحولات اساسی در دنیای کامپیوتر شد،** 8088 **بود. ریزپردازنده فوق در سال** 1979 **توسط شرکت** IBM **طراحی و اولین نمونه آن در سال** 1982 **عرضه گردید. وضعیت تولید ریزپردازنده توسط شرکت های تولید کننده بسرعت رشد و از مدل** 8088 **به** 80286، 80386، 80486، **پنتیوم، پنتیوم** II، **پنتیوم** III **و پنتیوم** 4 **رسیده است. تمام پردازنده های فوق توسط شرکت اینتل و سایر شرکت های ذیربط طراحی و عرضه شده است. پردازنده های پنتیوم** 4 **در مقایسه با پردازنده** 8088 **عملیات مربوطه را با سرعتی به میزان** 5000 **بار سریعتر انجام می دهد! جدول زیر ویژگی هر یک از پردازنده های فوق بهمراه تفاوت های موجود را نشان می دهد.**

MIPS	Data width	Clock speed	Microns	Transistors	Date	Name
0.64	8 bits	2 MHz	6	6,000	1974	8080
0.33	16 bits 8-bit bus	5 MHz	3	29,000	1979	8088
1	16 bits	6 MHz	1.5	134,000	1982	80286
5	32 bits	16 MHz	1.5	275,000	1985	80386
20	32 bits	25 MHz	1	1,200,000	1989	80486
100	32 bits 64-bit bus	60 MHz	0.8	3,100,000	1993	Pentium
~300	32 bits 64-bit bus	233 MHz	0.35	7,500,000	1997	Pentium II
~510	32 bits 64-bit bus	450 MHz	0.25	9,500,000	1999	Pentium III
~1,700	32 bits 64-bit bus	1.5 GHz	0.18	42,000,000	2000	Pentium 4

توضیحات

- **ستون** *Date* **نشاندهنده سال عرضه پردازنده است.**
- **ستون** *Transistors* **تعدا ترانزیستور موجود بر روی تراشه را مشخص می کند. تعداد ترانزیستور بر روی تراشه در سال های اخیر شتاب بیشتری پیدا کرده است.**
- **ستون** *Micron* **ضخامت کوچکترین رشته بر روی تراشه را بر حسب میکرون مشخص می کند (ضخامت موی انسان** 100 **میکرون است).**
- **ستون** *Speed Clock* **حداکثر سرعت** Clock **تراشه را مشخص می نماید.**
- **ستون** *Width Data* **پهنای باند واحد منطق و محاسبات** (ALU) **را نشان می دهد. یک واحد منطق و حساب هشت بیتی قادر به انجام عملیات محاسباتی نظیر: جمع، تفریق، ضرب و ... برای اعداد هشت بیتی است. در صورتیکه یک واحد منطق و حساب** 32 **بیتی قادر به انجام عملیات بر روی اعداد** 32 **بیتی است. یک واحد منطق و حساب** 8 **بیتی به منظور جمع دو عدد** 32 **بیتی می بایست چهار دستورالعمل را انجام داده در صورتیکه یک واحد منطق وحساب** 32 **بیتی**

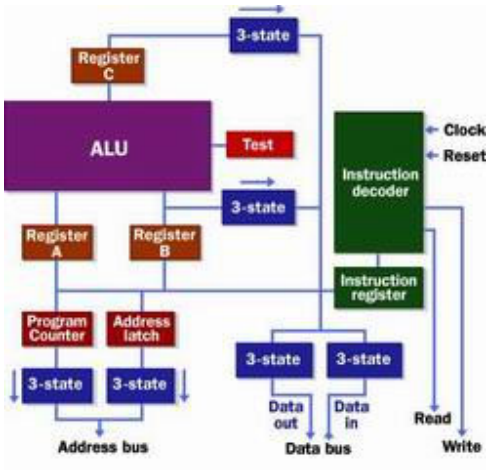
عملیات فوق را صرفا" با اجرای یک دستورالعمل انجام خواهد داد.در اغلب موارد گذرگاه خارجی داده ها مشابه ALU **است. وضعیت فوق در تمام موارد صادق نخواهد بود مثلا" پردازنده** 8088 **دارای واحد منطق وحساب** 16 **بیتی بوده در حالیکه گذرگاه داده ئی آن هشت بیتی است. در اغلب پردازنده های پنتیوم جدید گذرگاه داده** 64 **بیتی و واحد منطق وحساب** 32 **بیتی است. ستون** *MIPS* **مخفف کلمات** per Second Millions of instruction **(میلیون دستورالعمل در هر ثانیه) بوده و واحدی برای سنجش کارآئی یک پردازنده است.**

درون یک پردازنده

به منظورآشنائی با نحوه عملکرد پردازنده لازم است، نگاهی به درون یک ریزپردازنده داشته و با منطق نحوه انجام عملیات بیشتر آشنا شویم. یک ریزپردازنده مجموعه ای از دستورالعمل ها را اجراء می کند. دستورالعمل های فوق ماهیت و نوع عملیات مورد نظر را برای پردازنده مشخص خواهند کرد. با توجه به نوع دستورالعمل ها، یک ریزپردازنده سه عملیات اساسی را انجام خواهد داد:

- **یک ریزپردازنده با استفاده از واحد منطق و حساب خود (ALU) قادر به انجام عملیات محاسباتی نظیر: جمع، تفریق، ضرب و تقسیم است. پردازنده های جدید دارای پردازنده های اختصاصی برای انجام عملیات مربوط به اعداد اعشاری می باشند.**
- **یک ریزپردازنده قادر به انتقال داده از یک محل حافظه به محل دیگر است.**
- **یک ریزپردازنده قادر به اتخاذ تصمیم (تصمیم گیری) و پرش به یک محل دیگر برای اجرای دستورالعمل های مربوطه بر اساس تصمیم اتخاذ شده است.**

شکل زیر یک پردازنده ساده را نشان می دهد.



پردازنده صفحه قبل دارای:

- **یک گذرگاه آدرس** (Address Bus) **است که قادر به ارسال یک آدرس به حافظه است (گذرگاه فوق می تواند** 8 ، 16 **و یا** 32 **بیتی باشد).**
- **یک گذرگاه داده** (Data Bus) **است که قادر به ارسال داده به حافظه و یا دریافت داده از حافظه است (گذرگاه فوق می تواند** 8 ، 16 **و یا** 32 **بیتی باشد).**
- **یک خط برای خواندن** (RD) **و یک خط برای نوشتن** (WR) **است که آدرسی دهی حافظه را انجام می دهند. آیا قصد نوشتن در یک آدرس خاص وجود داشته و یا مقصود، خواندن اطلاعات از یک آدرس خاص حافظه است؟**
- **یک خط** Clock **که ضربان پردازنده را تنظیم خواهد کرد.**
- **یک خط** Reset **که مقدار "شمارنده برنامه" را صفر نموده و یا باعث اجرای مجدد یک فرآیند می گردد.**

فرض کنید پردازنده فوق هشت بیتی بوده واز عناصر زیر تشکیل شده است:

- **ریجسترهای** C, B, A **نگاهدارنده هائی بوده که از فلیپ فلاپ ها ساخته شده اند.**
- Latch Address **مشابه ریجسترهای** C, B, A **است.**
- **شمارنده برنامه** (Program Counter) **نوع خاصی از یک نگهدارنده اطلاعات است که قابلیت افزایش بمیزان یک و یا پذیرش مقدار صفر را دارا است.**
- **واحد منطق و حساب** (ALU) **می تواند یک مدار ساده جمع کننده هشت بیتی بوده و یا مداری است که قابلیت انجام عملیات جمع، تفریق، ضرب و تقسیم را دارا است.**
- **ریجستر** Test **یک نوع خاص نگاهدارنده بوده که قادر به نگهداری نتایج حاصل از انجام مقایسه ها توسط** ALU **است.** ALU **قادر به مقایسه دو عدد وتشخیص مساوی و یا نامساوی بودن آنها است. ریجستر** Test **همچنین قادر به نگهداری یک**Carry bit **(ماحصل آخرین مرحله عملیات جمع) است. ریجستر فوق مقادیر مورد نظر را در فلیپ فلاپ ها ذخیره و در ادامه** Instruction Decoder "**تشخیص دهنده دستورالعمل ها" با استفاده از مقادیر فوق قادر به اتخاذ تصمیمات لازم خواهد بود.**
- **همانگونه که در شکل فوق، مشاهده می گردد از شش** "State–3" **استفاده شده که به آنها** "tri-State buffers" **می گویند. بافرهای فوق قادر به پاس دادن مقادیر صفر و یا یک و یا قطع خروجی مربوطه می باشند. این نوع بافرها امکان ارتباط چندین خروجی را از طریق یک** Wire

فراهم می نمایند. در چنین حالتی فقط یکی از آنها قادر به انتقال (حرکت) صفر و یا یک بر روی خط خواهد بود.

ریجستر Instruction **و** Instruction Decoder **مسئولیت کنترل سایر عناصر را برعهده خواهند داشت. بدین منظور از خطوط کنترلی متفاوتی استفاده می گردد. خطوط فوق در شکل فوق نشان داده نشده اند ولی می بایست قادر به انجام عملیات زیر باشند:**

- **به ریجستر** A **اعلام نماید که مقدار موجود بر روی گذرگاه داده را در خود نگاهدارد** (Latch)**.**
- **به ریجستر** B **اعلام نماید که مقدار موجود بر روی گذرگاه داده را در خود نگاهدارد** (Latch)**.**
- **به ریجستر** C **اعلام نماید که مقدار موجود بر روی گذرگاه داده را در خود نگاهدارد** (Latch)**.**
- **به "شمارنده برنامه" اعلام نماید که مقدار موجود بر روی گذرگاه داده را در خود نگاهدارد** (Latch)**.**
- **به ریجستر** Address **اعلام نماید که مقدار موجود بر روی گذرگاه داده را در خود نگاهدارد** (Latch)**.**
- **به ریجستر** Instruction **اعلام نماید که مقدار موجود بر روی گذرگاه داده را در خود نگاهدارد** (Latch)**.**
- **به "شمارنده برنامه" اعلام نماید که مقدار خود را افزایش دهد.**
- **به "شمارنده برنامه" اعلام نماید که مقدار خود را صفر** (Reset) **نماید.**
- **به واحد منطق و حساب نوع عملیاتی را که می بایست انجام گیرد، اعلام نماید.**
- **به ریجستر** Test **اعلام نماید که بیت های ماحصل عملیات** ALU **را در خود نگاهدارد.**
- **فعال نمودن خط** RD **(خواندن).**
- **فعال نمودن خط** WR **(نوشتن).**

حافظه های RAM و ROM

در بخش قبل گذرگاه های آدرس و داده نظیرخطوط WR, RD **بررسی گردیدند. گذرگاه های فوق به حافظه های** RAM، ROM **و یا هر دو متصل خواهند بود. در ریزپردازنده ساده فرضی فوق، از گذرگاه های آدرس و داده هشت بیتی استفاده می گردد. بدین ترتیب پردازنده قادر به آدرس دهی** 256 **بایت حافظه و خواندن و یا نوشتن هشت بیت از حافظه در هر لحظه خواهد بود. فرض کنید پردازنده فوق دارای** 128 **بایت حافظه** ROM **بوده که از آدرس صفر شروع شده و** 128 **بایت حافظه** RAM **که از آدرس** 128 **آغاز می گردد، است. حافظه** ROM **تراشه ای است که اطلاعاتی را از قبل و بصورت دائم در**

خود نگهداری می نماید. گذرگاه های آدرس به تراشه ROM اعلام خواهند کرد که کدام بایت را خواسته و آن را بر روی گذرگاه قرار خواهد داد. زمانیکه وضعیت خط RD تغییر نماید تراشه ROM بایت مورد نظر و انتخابی را بر روی گذرگاه داده قرار خواهد داد. RAM شامل بایت هائی از اطلاعات است. ریزپردازنده قادر به خواندن و نوشتن در حافظه فوق بر اساس سیگنال های دریافتی از خطوط RD و RW است. در رابطه با حافظه RAM می بایست به این نکته نیز اشاره گردد که این نوع از حافظه ها با از دست منبع انرژی (برق) اطلاعات خود را از دست خواهند داد.

تمامی کامپیوترها دارای حافظه ROM به میزان مشخص می باشند (برخی از کامپیوترها ممکن است دارای حافظه RAM نبوده نظیر میکرو کنترل ها، ولی وجود و ضرورت حافظه ROM را در هیچ کامپیوتری نمی توان انکار نمود). بر روی کامپیوترهای شخصی حافظه ROM را BIOS نیز می نامند. زمانیکه ریزپردازنده فعالیت خود را آغاز می نماید، در ابتدا دستورالعمل هائی را اجراء خواهد کرد که در BIOS می باشند. دستورالعمل های موجود در BIOS عملیاتی نظیر تست سخت افزار و سیستم را انجام و در ادامه فرآیندی آغاز خواهد شد که نتیجه آن استقرار سیستم عامل در حافظه خواهد بود (Booting). در آغاز فرآیند فوق، بوت سکتور هارد دیسک (می تواند آغاز عملیات فوق از هارد شروع نشده و از فلاپی دیسک انجام گردد، اتخاذ تصمیم در رابطه با وضعیت فوق بر اساس پارامترهای ذخیره شده در حافظه CMOS خواهند بود) را بررسی خواهد کرد. بوت سکتور فوق حاوی برنامه ای کوچک است که در ادامه BIOS آن را خوانده و در حافظه RAM مستقر خواهد کرد. ریزپردازنده در ادامه دستورالعمل های مربوط به برنامه بوت سکتور را که در حافظه RAM مستقر شده اند ،اجراء خواهد کرد. برنامه فوق به ریزپردازنده اعلام خواهد کرد که اطلاعات دیگری را از هارد دیسک به درون حافظه RAM انتقال و آنها را اجراء نماید. با ادامه وتکمیل فرآیند فوق سیستم عامل در حافظه مستقر ومدیریت خود را آغاز می نماید.

دستورالعمل های ریزپردازنده

هر ریزپردازنده دارای مجمو عه ای از دستورالعمل ها بوده که دارای کارآئی خاصی می باشند. این دستورالعمل ها بصورت الگوئی از صفر و یا یک پیاده سازی می گردنند. استفاده از دستورات فوق با توجه به ماهیت الگوئی آنها برای انسان مشکل و بخاطر سپردن آنها امری است مشکل تر!، بدین دلیل از مجموعه ای "کلمات" برا ی مشخص نمودن الگوهای فوق استفاده می گردد. مجموعه "کلمات" فوق "زبان اسمبلی" نامیده می شوند. یک "اسمبلر" قادر به ترجمه کلمات به الگوهای بیتی متناظر است. پس از ترجمه، ماحصل عملیات که همان استخراج "الگوهای بیتی" است، در حافظه مستقر تا زمینه اجرای

آنها توسط ریزپردازنده فراهم گردد جدول زیر برخی از دستورالعمل های مورد نیاز در رابطه با پردازنده فرضی را نشانن می دهد.

Meaning	Instruction
لود نمودن ریجستر A از آدرس حافظه	LOADA mem
لود نمودن ریجستر B از آدرس حافظه	LOADB mem
لود نمودن یک مقدار ثابت در ریجستر B	CONB con
ذخیره نمودن مقدار موجود در ریجستر B در یک آدرس حافظه	SAVEB mem
ذخیره نمودن مقدار موجود در ریجستر C در یک آدرس حافظه	SAVEC mem
جمع A و B و ذخیره کردن حاصل در C	ADD
تفریق A و B و ذخیره کردن حاصل در C	SUB
ضرب A و B و ذخیره کردن حاصل در C	MUL
تقسیم A و B و ذخیره کردن حاصل در C	DIV
مقا یسه A و B و ذخیره کردن حاصل در Test	COM
پرش به یک آدرس مشخص	JUMP addr
پرش شرطی (اگر مساوی است) به یک آدرس مشخص	JEQ addr
پرش شرطی (اگر نا مساوی است) به یک آدرس مشخص	JNEQ addr
پرش شرطی (اگر بزرگتر است) به یک آدرس مشخص	JG addr
پرش شرطی (اگر بزرگتر و یا مساوی است) به یک آدرس مشخص	JGE addr
پرش شرطی (اگر کوچکتر است) به یک آدرس مشخص	JL addr
پرش شرطی (اگر کوچکتر و یا مساوی است) به یک آدرس مشخص	JLE addr
توقف اجراء	STOP

مثال: فرض کنید برنامه محاسبه فاکتوریل عدد پنج (5!=5*4*3*2*1) با یکی از زبانهای سطح بالا نظیر C نوشته گردد. کمپایلر (مترجم) زبان C برنامه مورد نظر را به زبان اسمبلی ترجمه خواهد کرد (فرض کنید که آدرس شروع RAM در پردازنده فرضی 128 و آدرس شروع حافظه ROM صفر باشد). جدول زیر برنامه نوشته شده به زبان C را بهمراه کد ترجمه شده اسمبلی معادل آن، نشان می دهد.

Assembly Language	C Program
// Assume a is at address 128	
// Assume F is at address 129	
0 CONB 1 // a=1;	
1 SAVEB 128	
2 CONB 1 // f=1;	
3 SAVEB 129	
4 LOADA 128 // if a > 5 the jump to	
17	a=1;
5 CONB 5	f=1;
6 COM	while (a <= 5)
7 JG 17	{
8 LOADA 129 // f=f*a;	f = f * a;
9 LOADB 128	a = a + 1;
10 MUL	}
11 SAVEC 129	
12 LOADA 128 // a=a+1;	
13 CONB 1	
14 ADD	
15 SAVEC 128	
16 JUMP 4 // loop back to if	
17 STOP	

در ادامه می بایست کدهای ترجمه شده به زبان اسمبلی به زبان ماشین (الگوهای بیتی) ترجمه گردند. بدین منظور لازم است که هر یک از دستورات اسمبلی دارای کد معادل (OpCode) باشند. فرض کنید دستورات اسمبلی در پردازنده فرضی دارای Opcode (کدهای عملیاتی) زیر باشند.

Opcode	Assembly Instruction
1	LOADA mem
2	LOADB mem
3	CONB con
4	SAVEB mem
5	SAVEC mem
6	ADD
7	SUB
8	MUL
9	DIV
10	COM
11	JUMP addr
12	JEQ addr
13	JNEQ addr
14	JG addr
15	JGE addr
16	JL addr
17	JLE addr
18	STOP

در نهایت برنامه ترجمه شده به زبان اسمبلی به زبان ماشین (الگوهای بیتی) ترجمه خواهد شد.

Bit Patterns	Assembly Language

// Assume a is at address 128
// Assume F is at address 129
Addr opcode/value
0 3 // CONB 1
1 1
2 4 // SAVEB 128
3 128
4 3 // CONB 1
5 1
6 4 // SAVEB 129
7 129
	// Assume a is at address 128
	// Assume F is at address 129
	0 CONB 1 // a=1;
	1 SAVEB 128
	2 CONB 1 // f=1;
	3 SAVEB 129
	4 LOADA 128 // if a > 5 the jump to 17
	5 CONB 5
	6 COM
	7 JG 17

8	1	// LOADA 128	8	LOADA 129	// f=f*a;
9	128		9	LOADB	128
10	3	// CONB 5	10		MUL
11	5		11	SAVEC	129
12	10	// COM	12	LOADA 128	// a=a+1;
13	14	// JG 17	13	CONB	1
14	31		14		ADD
15	1	// LOADA 129	15	SAVEC	128
16	129		16	JUMP 4	// loop back to if
17	2	// LOADB 128	17	STOP	
18	128				
19	8	// MUL			
20	5	// SAVEC 129			
21	129				
22	1	// LOADA 128			
23	128				
24	3	// CONB 1			
25	1				
26	6	// ADD			
27	5	// SAVEC 128			
28	128				
29	11	// JUMP 4			
30	8				
31	18	// STOP			

همانگونه که مشاهده می نمائید برنامه نوشته شده به زبان C **به** 17 **دستورالعمل معادل اسمبلی و** 31 **دستورالعمل زبان ماشین تبدیل گردید.**

Instruction Decoder **(تشخیص دهنده نوع دستورالعمل ها) با انجام عملیاتی خاص، نوع دستورالعمل را تشخیص خواهد داد. فرض کنید دستور العمل** ADD **را داشته باشیم و بخواهیم نحوه تشخیص دستورالعمل را دنبال نمائیم:**

- **در زمان اولین** Clock**، دستورالعمل** Load **می گردد (فعال کردن بافر** tri-state **برای "شمارنده برنامه "، فعال شدن خط** RD**، فعال کردن** Data-in **در بافر** tri-state**).**
- **در زمان دومین** Clock**، دستورالعمل** ADD **تشخیص داده خواهد شد (تنظیم عملیات جمع برای** ALU **، ذخیره نمودن ماحصل عملیات** ALU **در ریجستر** C**).**

- **در زمان سومین** Clock**، "شمارنده برنامه" افزایش خواهد یافت (در تئوری این مرحله می تواند در زمان دومین** Clock **نیز صورت پذیرد).**

همانگونه که ملاحظه گردید، هر دستورالعمل اسمبلی دارای چندین Clock Cycle **است. برخی از دستورات نظیر** ADD **دارای دو و یا سه** Clock **و برخی دیگر از دستورات دارای پنج ویا شش** Clock **خواهند بود.**