دانشگاه آزاد اسلامی واحد ابهر

گروه کارشناسی ارشد تاریخ

پایان نامه درجه کارشناسی ارشد تاریخ M.A

عنوان :

سیمای سیاسی و اجتماعی خواجه رشید الدین فضل الله همدانی

استاد مربوطه:

دکتر محمود خامچی

استاد مشاور:

دکتر نصرت الله بیات

تهیه کننده :

محمد رضا صفری

شهریور ماه
۱۳۸۵

Microprocessor

۱- مقدمه ای بر ریزپرازنده

۲- ساختار عمومی ریزپردازنده های پیشرفته

۳- معماری ریزپردازنده؟

- مجموعه دستورالعمل های ریزپردازنده

- قالب داده ها

- قالب برای دستورالعمل ها

- شیوه های آدرس دهی

۴- سلسله مراتب حافظه

- رجیستر فایل

- Cache

- حافظه مجازی و صفحه بندی

- قطعه بندی

- جداول صفحه، TBL، حفاظت

۵- پایپ لاین

- خط لوله دستورالعمل

- مخابرات خط لوله

- پیشگویی انشعاب

- Bypassing یا Result Forwarding

۶- IPL (Instruction Level Parallelism)

- ILP چیست؟

- محدودتی های ILP

- پردازنده های سوپر اسکالر

- اجرای ترتیبی و خارج از نوبت

- Register renaming

- پردازنده های VLIW

- تکنیک های کامپایل برای ILP

۷- اصول کامپیوترهای RISC

- RISC در برابر CISC

- ارزیابی سیستم های RISC

- ارزیابی Cache در مقابل رجیستر فایل تراشه

۸- I/O

- گذرگاه های I/O

- وقفه ها

- Memory Mapped I/O

- DMA

- دستگاه های I/O

- سیستم های دیسک

۹- چند پردازنده ای

- speedup و کارآیی

- سیستم های چند پردازنده ای

- سیستم های Message passing

- سیستم های Shared Memory

Case Study

خانواده اینتل

- معماری خانواده اینتل

- مجموعه ثبات:

- قالب های داده

- شیوه های آدرس دهی

- وقفه

- قطعه بندی

- صفحه بندی

- مکانیزم حفاظت

- پنتیوم

- ریزپردازنده های i486 و i386

- 8086 و 80186 و 80286

- 8086 و 8088

خانواده موتورولا M68000

- ثبات های پردازنده

- قالب های داده

- شیوه های آدرس دهی

- مجموعه دستورالعمل ها

- مدیریت حافظه

ریزپردازنده های پیشرفته RISC

- پردازنده DEC Alpha AXP

- معماری آلفا

خانواده Power PC

- معماری Power PC

- IBM RS/6000

خانواده Sparc

- معماری اسپارک

- سوپر اسپارک

خانواده MIPS RX000

- معماری MIPS

- R4400 و MIPS R4000

- خانواده اینتل i860

خانواده موتورولا M88000

- معماری M88000

- معماری MC88110

معماری HP

- معماری

- حافظه

چکیده

در اواسط دهه 70 ریزپردازنده ها ساختار ساده ای داشتند و در این زمان هر ریزپردازنده از یک واحد پردازشگر مرکزی (cpu) و یک تراشه LSI (شامل 5/000 ترازیستور) تشکیل شده بود و با فرکانس 1 تا 5 مگاهرتز در یک سیستم 8 بیتی کار می کرد و این ریزپردازنده ها دارای 2 الی 7 ثبات 8 بیتی بودند. به خاطر قیمت و بهای اندک و اندازه کوچک ریزپردازنده ها، در بیشتر سیستم های کامپیوتری از آنها استفاده می شد و به جائی رسید که جایگزین سیستم های

mainframe و میکروکامپیوترها شدند. با ظهور ریزپردازنده ها هر خانه ای دارای یک کامپیوتر دیجیتالی است.

از دهه 70 به بعد ریزپردازنده ها تغییرات زیادی کرده اند و در دهه 90 ریزپردازنده ها 32 بیتی تا 64 بیتی شدند. و با فرکانس هایی از 25 تا 200MHZ کار می کردند و عملاً دارای تراشه هایی با سه میلیون ترازیستور بودند (VLSI). اکثر این ریزپردازنده ها قادر بودند بیشتر از یک دستورالعمل را در یک چرخه اجرا کنند. تمامی ریزپردازنده‌های پیشرفته دارای یک تراشه FPU هستند و اکثر آن ها دارای 16تا 32 ثبات همه منظوره در CPU و یک رجیستر فایل با 32 ثبات برای IU و یک رجیستر فایل با 32 ثبات جداگانه برای FPU هستند.

خیلی از ریزپردازنده ها برای عملیات شناور و عملیات صحیح دارای Operational Unit هستند و مقدار قابل توجهی Cache دارند. در اکثر آنها Cache شامل Cache داده و Cache دستورالعمل است . کارآیی ریزپردازنده های پیشرفته امروز مساوی یا بیشتر از Mainframe و یا سوپر کامپیوترهای دوران قبل می باشد.

تعداد زیادی کارخانه سازنده ریزپردازنده وجود دارد که دارای ویژگی های خاص خود می باشند و دو گروه از گسترده ترین خانواده ریزپردازنده که در دهه 70 ساخته شده اند عبارتند از اینتل X86 یا 80X86 و خانواده موتورولا M680X0.

نزدیک به دهه 80 ما شاهد یک توسعه موازی روی معماری های جدید بوده ایم که تمایل به کامپیوترهائی با مجموعه دستورالعمل کاهش یافته یا RISC بوده اند. خانواده های اینتل X86 و موتورولا M68000 از کلاس غیرRISC یعنی کامپیوترهایی با مجموعه دستورهای پیچیده یا CISC تشکیل شده اند.

اینتل، یک ریزپردازنده ۴ بیتی به نام 4004 در سال 1971 شروع کرد که در یک ماشین حساب معمولی بکار میرفت، و به آسانی محاسبات BCD را انجام میداد. در سال 1972 ریزپردازنده 8 بیتی 8008 توسعه یافت و در سال 1974 یک ریزپردازنده قدرتمند 8 بیتی به نام 8080 تولد یافت و به دنبال آن 8085 در سال 1976 به بازار آمد. بخشی از معماری 8085/8080 همانند مجموعه ثبات ها در خانواده X86 همچنان استفاده می شود. اینتل ساخت ریزپردازنده های خانواده X86 را با یک ریزپردازنده 16 بیتی به نام 8086 در 1978 آغاز کرد و عملاً تمامی کارخانه های معروف ریزپردازنده های 16بیتی بعد از دهه 70 و تا نزدیک دهه 80 از یک تراشه ارزان قیمت و یک گذرگاه خارجی 8 بیتی با یک معماری داخلی 16 بیتی استفاده می کردند.

در سال 1969 ریزپردازنده 8080 با باس خارجی 8 بیتی و گذرگاه داخلی 16 بیتی ایجاد شد و برای گذرگاه داده خود 50% به تراشه های میانی کمتری در مقایسه با پردازنده های 16 بیتی نیاز داشتند و هزینه آن ها نیز کمتر بود.

در واقع 8080 به پردازنده اصلی شرکت IBM روی کامپیوتر های شخصی (PC) تبدیل و در تمامی تولیدات بعدی در جهان منتشر گردید. به دنبال 8086، تکامل یافته آن یعنی 80186 ساخته شد که همان 8086 همراه با تعدادی اینترفیس I/O و واحدهای منطقی وی یک تراشه بود و تعداد کمی دستورالعمل به آن اضافه شده بود.

وقتی 80286 به عنوان یک ریزپردازنده 16 بیتی در سال 1982 شناخته شد، مرحله جدیدی در توسعه ریزپردازنده ها پدید آمد که قابلیت Protected Mode نامیده شد. این شیوه در تمامی محصولات دیگر خانواده اینتل بکار گرفته شد.

اولین پردازنده 32 بیتی اینتل در سال 1985 بود که i386 نام داشت. اینتل یک پردازنده 32 بیتی به نام 432 که با خانواده X86 ناسازگار بود را زودتر از دهه 80 بیرون داده بود که هرگز تجاری نشد. در سال 1989 ریزپردازنده i486 توسعه یافت که سرعت عملکرد آن در مدل DX2 به 66MHZ میرسید. i486 دارای یک واحد FPU و یک Cache به اندازه 8KB در داخل تراشه است.

محصول بعدی خانواده اینتل پنتیوم بود که در سال 1993 ساخته شد و قبل از گسترش به آن i586 یا P5 گفته می شد اما اینتل تصمیم گرفت نام آن را پنتیوم بگذارد. پنتیوم یک سوپر اسکالر دو سطحی است یعنی دو دستورالعمل را موازی واکشی و کدگشایی و اجرا می کند و دارای

گذرگاه داخلی 64 بیتی و یک cache به اندازه 16KB است (8K data cache + 8K inst.cache).

اینتل مرتباً روی محصولات بعدی کار می کرد و P6 نامیده شد که طبق وعده آن کارآیی پنتیوم را حد اقل 2 برابر کرد و پس از آن به ترتیب Pentium II، Pentium III و Pentium IV نیز ساخته شدند.

توسعه ریزپردازنده های موتورولا نیز شبیه اینتل است. موتورولا در سال 1974 خانواده 8 بیتی 6800 را ایجاد کرد که در سال 1977 به یک محصول 8 بیتی ویژه به نام 6809 تبدیل شد. موتورولا نخستین ریزپردزنده 16 بیتی خود را در سال 1979 به نام M68000 به بازار عرضه کرد. از ویژگی های مهم خانواده M68000 آن است که ریزپردازنده MC68000 به طور پایه ای یک سیستم 16 بیتی با گذرگاه داده 16 بیتی و 16 ثبات پردازنده UPRS و شمارنده برنامه 32 بیتی است. در سال 1984 ریزپردازنده 32 بیتی MC60000 نامیده شد. هم چنین موتورولا روایت های توسعه یافته محصولات MC68000 را تحت عنوان MC68010 .و MC68012 ایجاد کرد. MC68010 از نظر پایه ها کاملاً با MC68000 سازگار بود بنابراین از نظر طراحی سیستم، یک MC68000 می توانست با یک MC68010 جایگزین شود.

موتورولا ساخت ریزپردازنده های 32 بیتی خانواده M68000 را در سال 1984 با نام MC68020 شروع کرد. این ریزپردازنده ها در سال 1978 به ریزپردازنده MC68030 و در

سال 1989 به MC68040 توسعه پیدا کردند. 68020 دارای یک Cache دستور کوچک 256 بایتی در داخل تراشه بود. 68030 دارای یک حافظه Cache دستور کوچک 256 بایتی در داخل تراشه بود. 68030 دارای یک حافظه Cache دوگانه نسبتاً کوچک (256 data + 256 B inst.) و 68040 دارای Cache 4KB دستور و Cache 4KB داده در یک تراشه می باشد.

عضو بعدی خانواده M68000 که در سال 1994 ساخته شد MC68060 نام گرفت که یک سوپر اسکالر ۲ سطحی با Cache 8KB دستور و Cache داده در تراشه بود و دستورالعمل ها با سرعت 66MHZ کار می کرد.

البته چندین خانواده از ریزپردازنده ها وجود دارد که به آن ها اشاره اندکی خواهد شد، مانند Z-80 مربوط به شرکت Zilog که ریزپردازنده معروف 8 بیتی است و توسط گروهی از طراحان متخصص اینتل که 8080 و 8085 را ساخته بودند ایجاد شد و از اینرو Z-80 شبیه معماری اینتل 8080 است ولی زبان اسمبلی هر دو متفاوت می باشد و Z-80 دارای دستورهای بیشتری نسبت به 8080 است. شرکت Zilog خانواده 16 بیتی خود را با نام Z8000 موازی با اینتل 8086 منتشر کرد و Z80000 یک محصول 32 بود که به تولید نرسید.

توسعه ریزپردازنده های RISC به شکل موازی با اینتل و موتورولا انجام شد. اولین مستندات RISC در دانشگاه برکلی در سال 1980 توسط Pattersonو Ditzel ارایه شد و لغت RISC

توسط کارلو اسکوئین لرپرت اولین مقاله RISC عنوان شد. طراحیس سیستم محاسباتی از نوع RISC بدون استفاده از واژه RISC خیلی پیشتر توسط IBM در دهه 70 آغاز شد که با نام IBM801 شناخته می شد. هم چنین دانشگاه استنفورد در سال 1981 سیستم RISC خود را ایجاد کرد و به نام MIPS (Microprocessor without interlocked Pipeline Stage) نامید.

شرکت MIPS روایت های تجاری را تحت خانواده RX000 (X=6,4,3,2) منتشر ساخته است. در برخی از خصوصیات اولیه طرح RISC برکلی در Sparc از شرکت Sun Micro System به کار رفته است. شرکت اینتل خانواده RISC خود را با نام I860 و شرکت موتورولا با نام M88000 شروع کردند. شرکت IBM کامپیوتر RISC خود را با نام ROMP (Research Office Product Division Microprocessor) ادامه داد و به سیستم RS6000 منجر شد. معماری RS6000 با خصوصیت جدید ریزپذدازنده های IBM و موتورولا و اپل ترکیب شد و Power PC ساخته گردید. اولین محصول 601 است که در 1991 منتشر شد و به دنبال آن 620 با توانائیهای بیشتری ایجاد گردید.

Intel 8086

در این فصل معماری داخلی، شیوه های آدرس دهی، مجموعه دستورالعمل ها و تکنیک های I/O مرتبط با ریزپردازنده 8086 را معرفی خواهیم کرد.

مقدمه

8086 اولین ریزپردازنده 16 بیتی اینتل بود. طراحی آن بر اساس 8080 ولی مستقیماً با آن سازگار نبود. 8086 با استفاده از فناوری HMOS طراحی شده بود و شامل 29000 ترانزیستور بود. 8086 در یک بسته 40 پایه ای قرار داشت و با منبع تغذیه 5V+ کار می کرد. 8086 در سه مد و سرعت مختلف Clock کار می کند. 8086 استاندارد با فرکانس کلاک داخلی 5MHZ و 8086-2 و 8086-4 با فرکانس های داخلی به ترتیب 8 , 4MHZ کار می کنند. به همین دلیل از یک مولد / درایور تراشه مثل Intel 8284 برای تولید سیگنال ورودی کلاک 8086 استفاده می شود. 8086 کلاک خارجی در پایه CLK را به 3 تقسیم می کند. این جمله به این معنی است که برای کلاک داخلی 5MHZ، 8284 باید کلاک خروجی 15MHZ را تولید کند تا به پایه CLK تراشه 8086 متصل شود.

8086 دارای آدرس 20 بیتی است و از اینرو می توان حداکثر تا 1MB dh 2^{20} از حافظه را آدرس دهی کند. 8086 از حافظه قطعه بندی شده استفاده می کند. نکته جالب توجه در 8086 این است که می تواند حداکثر تا 6 بایت دستورالعمل را از حافظه پیش واکشی (PreFetch) کند و آن ها را به ترتیب به صف کند و بدین ترتیب سرعت اجرای دستورالعمل را بهبود بخشد.

حافظه ریزپردازنده بر اساس 8086 برحسب بایت پیکره بندی می شود. هر بایت می تواند منحصراً با آدرس های 20 بیتی $00000_{16} \rightarrow FFFFF_{16}$ آدرس دهی شود. هر کلمه 16 بیتی 8086

شامل هر دو بایت مجاور یکدیگر است. بایت آدرس کمتر (L.O.B) و بایت آدرس بیشتر (H.O.B) به شکل زیر می باشند:

بایت با ارزش بیشتر	بایت با ارزش کمتر
26H	07H
آدرس 00521_{16}	آدرس 00520_{16}

بنابراین کلمه 16 بیتی ذخیره شده در آدرس زوج 00520عبارت است از: 2607_{16}

اکنون کلمه ای را با آدرس فرد در نظر بگیرید:

H.O.B	L.O.B
3FH	05H
آدرس 01258_{16}	آدرس 01257_{16}

کلمه ذخیره شده در آدرس فرد 01257_{16} عباتست از 3F0SH

معماری 8086

همانطوری که در شکل می بینیم، ریزپردازنده 8086 از نظر داخلی به دو بخش تابعی جدا تقسیم می شود. این دو بخش واحد واسط گذرگاه (BIU) و واحد اجرا (EU) می باشند. واحد BIU دستورها را واکشی می کند، داده ها را از حافظه و پورت ها میخواند و داده ها را به حافظه

و پورت های I/O می نویسد. Eu دستورهایی را که اخیراً توسط BIU واکشی شده اند را اجرا می کند. وظایف واحدهای BIU و Eu از یکدیگر مستقل هستند. BIU پردازنده 8086 را به دنیای بیرون مرتبط می سازد. BIU تمامی عملیات گذرگاه خارجی را آماده می سازد. BIU شامل ثبات های سگمنت، اشاره گر به دستور (IP)، صف دستورالعمل و مدارهای کنترل گذرگاه و تولید آدرس می باشد و اعمالی چون واکشی و صف بندی دستورها و کنترل گذرگاه را انجام میدهد.

صف دستور BIU به شکل FIFO است و گروهی از ثبات هایی است که شامل ۶ بایت کد دستور واکشی شده از حافظه هستند. این امر با توجه به تسریع (Speed up)، اجرای برنامه را با همپوشانی (Over lupping) واکشی دستور با اجرا انجام میدهد. این مکانیزم را به عنوان پایپ لاین می شناسیم. چنانچه صف پر باشد و Eu برای دستیابی بحافظه درخواستی به BIU ندهد، BIU هیچ سیکل گذرگاهی را انجام نمی دهد. از طرف دیگر، اگر BIU پر نباشد و حداقل بتواند دو بایت را ذخیره سازد و Eu درخواستی را برای دستیابی بحافظه ندهد، BIU می تواند دستورالعمل ها را پیش واکشی کند. هرچند که اگر BIU برای دستیابی بحافظه توسط Eu وقفه داده شود درحالیکه BIU در حال پردازش واکشی یک دستورالعمل باشد، BIU ابتدا واکشی را کامل می کند و سپس به Eu سرویس می دهد. چنانچه دستوری مانند Jump یا Subroutine

Call داشته باشیم، BIU صف را reset میکند و پر کردن مجدد را پس از ارسال دستور جدید به Eu از نو شروع خواهد کرد.

همانطوری که می بینیم BIU شامل یک جمع کننده اختصاصی است که برای تولید آدرس 20 بیتی استفاده می شود. منطق کنترل گذرگاه BIU تمامی سیگنال های کنترل گذرگاه همچون سیگنال های خواندن و نوشتن را برای حافظه و I/O تولید می کند.

BIU دارای چهار ثبات سگمنت 16 بیتی است. این ثبات عبارتند از:

- ثبات سگمنت کد (CS)
- ثبات سگمنت داده (DS)
- ثبات سگمنت پشته (SS)
- ثبات فوق العاده (ES)

حافظه یک مگابایتی 8086 به سگمنتهایی حداکثر تا 64KB تقسیم می‌شود. 8086 می تواند مستقیماً چهار سگمنت را در یک زمان ویژه (256 کیلوبایت در حافظه یک مگابایتی) آدرس دهی کند. برنامه ها دستیابی به کد و داده را در سگمنت ها با تغییر محتویات ثبات سگمنت برای اشاره کرن به سگمنت های مطلوب، فراهم می سازد. تمامی دستورالعمل های جافظه باید در حافظه اصلی ذخیره شوند که توسط ثبات 16 بیتی CS و افست 16 بیتی در سگمنتی که در اشاره‌گر دستور (IP) 16 بیتی گنجانده شده به آن ااشره می شود. BIU

آدرس فیزیکی 20 بیتی داخلی را به وسیله آدرس منطقی فراهم شده توسط برنامه نویس (16 بیت CS و 16 بیت IP) تولید می کند. این کار با شیفت منطقی چهار بیتی CS به چپ و افزودن محتوای 16 بیتی IP تولید می شود. به عبارت دیگر CS توسط BIU برای تولید آدرس فیزیکی 20 بیتی در 16 ضرب می شود. به این معنی که تمامی دستورالعمل های برنامه نسبت به محتویات CS در 16 ضرب شده و سپس به آفست به دست آمده از IP 16 بیتی اضافه میشود.

دقت کنید که برای آدرس های کلمه، برنامه نویس از آدرس های مرتبه کمتر (زوج یا فرد) برای مشخص ساختن کلمه 16 بیتی استفاده می کند.

8086 همیشه به یک کلمه 16 بیتی به / از حافظه دستیابی پیدا می کند. 8086 می تواند در صورتیکه اولین بایت کلمه در آدرس زوج واقع شده باشد، در یک عملیات کلمه 16 بیتی را بخواند. به عبارت دیگر اگر اولین بایت کلمه یک آدرس فرد باشد، 8086 باید دو دستیابی به حافظه را برای خواندن دو بایت متوالی از حافظه انجام دهد. در این حالت 8086 از بایت یا بایت هایی که مورد نظرش نیست صرفنظر می کند. بطور مثال، دستور ADDR و MOV BX را در نظر بگیرید. دقت کنید وجود X (H یا L) به دنبال ثبات 8086 نشان دهندهٔ آن است که انتقال 16 بیتی یا 8 بیتی است.

این دستور محتوای مکان حافظه 20 بیتی آدرس دهی شده فیزیکی را که توسط ADDR آدرس دهی می شود به ثبات 16 بیتی BX منتقل می کند. اگر ADDR آدرس 20 بیتی زوج مثل 30024_{16} باشد دستور MOV محتوای BL را با محتوای خانه حافظه 30024_{16} و محتوی ثبات BH را با محتوی آدرس 30025_{16} در یک بار دستیابی پر می کند. اما اگر از ADDR در آدرس فرد مثلاً 40005_{16} باشد، دستور فوق BL را با محتوای 40005_{16} در و BH را با محتوی 40006_{16} در دو بار دستیابی حافظه Load می کند. دقت کنید 8086 به آدرس 40004_{16} و 40005_{16} دستیابی پیدا می کند ولی محتوی 40004_{16} را کنار می گذارد و نیز در عمل دوم به 40004_{16} و 40007_{16} دستیابی می کند ولی از محتوای 40007_{16} چشم پوشی می کند.

اکنون دستوری مانند MOV BH و ADDR را در نظر بگیرید. اگر آدرس ADDR زوج (مثل 50002) باشد، این دستور MOV به هر دو خانه 50002 و 50003 دستیابی می کند ولی BH را با محتوای 50002 پر کرده و از محتوی 50003 صرفنظر می کند. اما اگر ADDR در آدرس فرد باشد (مثل 5003) دستور MOV ثبات BH را با 50003 پر کرده و از 50002 صرفنظر می کند.

خانواده 8086 دارای دو نوع ریزپردازنده 16 بیتی است: 8086 و 8088. اختلاف مهم این است که پردازنده چگونه با دنیای بیرون ارتباط برقرار می کند. 8088 دارای یک مسیر داده (data path) 8 بیتی به حافظه I/O است ولی 8086 دارای یک مسیر داده خارجی 16

است. پس 8088 مجبور به دوبار عمل برای خواندن یک کلمه 16 بیتی در حافظه است. البته در بیشتر حالات این دو پردازنده مشابه هستند. از 8088 در طراحی کامپیوترهای شخصی IBM استفاده شده است.

8086 می تواند به شکل یک سیستم تک پردازنده کوچک (می نیمم مد در صورتیکه پایه $MN/\overline{MX}$ به high تنظیم شود) و یا به شکل یک سیستم چند پردازنده ای (ماکزیمم مد در صوتیکه پایه $MN/\overline{MX}$ به low تنظیم شود) پیکره بندی می گردد. در یک سیستم مفروض، پایه $MN/\overline{MX}$ دائماً به high یا low متصل می شود. برخی از پایه های 8086 دارای توابعی دوگانه بسته به نوع انتخاب سطح ولتاژ پایه $MN/\overline{MX}$ هستند. در مد می نیمم $MN/\overline{MX} = high$ این پایه ها سیگنال های کنترلی را مستقیماً به حافظه و وسایل I/O انتقال می دهند. از مد ماکزیمم $MN/\overline{MX} = low$ این پایه ها دارای وظایف متفاوتی برای تسهیلات سیستم های مالتی پروسسور است. در مد ماکزیمم توابع کنترلی که معمولاً در مد می نیمم نشان داده می شوند، فرض می شود که به وسیله تراشه ای مثل کنترل گذرگاه 8288 پشتیبانی می شوند.

پس از پیشرفت های فناوری، اینتل پردازنده ای پیشرفته تر 80186 و 80188 را که نسخه های پیشرفته تری از 8086 و 8088 بودند عرضه کرد. سرعت کلاک این ها 8MHZ بود که تقریباً دو برابر پردازنده های قبلی بودند. در روی این پردازنده‌های جدید از واحدهای جانبی مانند کنترل کننده DMA با تایمر 16 بیتی و واحد کنترل کننده وقفه استفاده می کنند.

شبیه 8086، 80186 دارای گذرگاه داده 16 بیتی و 80188 دارای گذرگاه داده 8 بیتی است و به غیر از این، معماری و مجموعه دستورهای 80186 و 80188 مشابه است. هم چنین 80186 و 80188 دارای یک مولد کلاک موجود در تراشه است که فقط از یک کریستال برای تولید کلاک استفاده می کند. مشابه 8085، فرکانس کلاک از نظر داخلی به 2 تقسیم می شود. یعنی کریستال خارجی 12 یا 16MHZ باید برای تولید فرکانس داخلی کلاک 6MHZ یا 8MHZ استفاده شود. 80186/80188 در یک بسته 68 پایه ای ساخته شده و هر دو پردازنده دارای مدارهای کنترل کننده اولویت وقفه برای فراهم ساختن 5 پایه وقفه هستند. مشابه 8086/8088 پردازنده های 80186/80188 می توانند مستقیماً 1MB از حافظه را آدرس دهی کنند. و دارای 10 دستورالعمب جدید نسبت به مجموعه دستورهای 8086/8088 هستند. مثلاً دستورهای IN و OUT برای گرفتن و یا نوشتن بایت یا کلمه رشته به کار می روند. از سوی دیگر 80286 دارای قابلیت های اضافی چون حفاظت (Protection) و مدیریت حافظه نسبت به معماری 8086 است. 80286 با سرعت 8MHZ دارای throughput تا 6 برابر در مقایسه با 8086 با سرعت 5MHZ است. 80286 در بسته 68 پایه ای ساخته شده است و می تواند در سرعتهای کلاک 4، 6 یا 8 مگاهرتز کار کند و به مولد کلاک خارجی 82284 برای تولید کلاک نیاز دارد. پردازنده 80286 کلاک خارجی را به 2 برای تولید کلاک داخلی تقسیم می کند و نوعاً در سیستم

های multiuser و multitasking استفاده می شود. 80286 در دو مد مختلف کار می کند. Real Addr و آدرس مجازی حفاظت شده.

در واقع آدرس فیزیکی تقلیدی از 8086 با کارآیی بالا است. در این مد 80286 مستقیماً تا 16 مگابایت حافظه را آدرس دهی کند. مد آدرس دهی مجازی، مدیریت حافظه مجازی، مدیریت Task و حفاظت را فراهم می سازد. برنامه نیس می تواند یکی از این دو مد را، با بارگذاری مناسب داده با دستورهای Load و Store در ثبات کلمه حالت ماشین (MSW) (Machine Status Word)که 16 بایتی است، انتخاب کند.

مثلاً دستورهای

Load MSW register = LMSW

Store MSW register = SMSW

80286 به عنوان CPU در کامپیوترهای شخصی IBM PC/AT به کار گرفته شد. نسخه های پیشرفته تری مانند 386 و 486 و ... بعداً بررسی خواهد شد.

مثال: $[CS] = 456A_{16}$

$[IP] = 1620_{16}$

20 بیت آدرس فیزیکی به شکل زیر محاسبه می شود

محتوای CS چهار بیت به یک شکل منطقی به چپ شیفت داده می شود $[CS] = 456A0_{16}$

$+[IP] = 1620_{16}$

$EA = 46CC0_{16}$ آدرس فیزیکی ۲۰ بیتی

BIU همیشه چهار صفر را در چهار بیت مرتبهٔ پائین آدرس فیزیکی (سگمنت) درج می کند. به عبارت دیگر CS شامل base یا store سگمنت کد جاری و IP شامل فاصله یا distanc از این آدرس به بایت دستور بعدی است کد مناسب واکشی شود. دقت کنید که داده های بلافاصله بعدی به عنوان بخشی از سگمنت کد در نظر گرفته می شوند.

ثبات SS به پشته جاری اشاره می کند. آدرس فیزیکی 20 بیتی پشته از ss و sp برای دستورهای پشته مانند push و pop محاسبه می شود. برنامه نویس می تواند از ثبات BP به عوض sp برای دستیابی به پشته در مد آدرس دهی پایه استفاده کند. در این حالت آدرس فیزیکی 20 بیتی از BP و ss محاسبه می شود.

ثبات DS به سگمنت داده جاری اشاره می کند. برای اغلب دستورها، عملوندها از این سگمنت واکشی می شوند. محتوی 16 بیتی SI (شاخص مبدأ) و DI (شاخص مقصد به عنوان آفست برای محاسبه آدرس فیزیکی 20 بیتی استفاده می شوند.

ثبات ES به سگمنت اضافی (Extra Segment) که داده‌ها ذخیره می‌شوند اشاره دارد (علاوه بر 64K اشاره شده توسط DS). دستورهای رشته معمولاً از ES و DI برای تعیین آدرس فیزیکی 20 بیتی مقصد استفاده می کنند.

سگمنت ها می توانند دارای چهار وضعیت باشند:

- Continious
- Partially Over lapped
- Fully overlapped
- Disjoint

شکل زیر پنج سگمنت (0# تا 4#) را نشان میدهد که در حافظه فیزیکی ذخیره شده اند:

نکته => هر سگمنت باید از مرزهای حافظه 16 بایتی آغاز شود.

مثالهایی از مقادیر سگمنت براساس آدرس‌فیزیکی شروع شده در 00000_{16}، 00000_{16}، 00020_{16} ، 00030_{16}، ... و $FFFF0_{16}$باشد. هر مکان حافظه فیزیکی می تواند به یک یا چند سگمنت منطقی Map شود.

یک سگمنت ممکن است با بیش از یک ثبات سگمنت اشاره دهی شود. مثلاً، DS و ES ممکن است به سگمنت مشابهی در حافظه اشاره کنند اگر رشته گنجانده شده در آن سگمنت به عنوان یک سگمنت Source در یک دستور رشته و به عنوان سگمنت مقصد در دستور رشته ای دیگر باشد. دقت کنید که در یک دستور رشته سگمنت مقصد باید توسط ES اشاره دهی شود.

قابل یادآوری است که کدها نباید در 6 بایت آخر حافظه فیزیکی نوشته شوند. نقص این مسأله ممکن است منجر به واکشی opcode از حافظه غیرموجود شوند و سبب hang کردن cpu گردد.

طراحی سیستم با استفاده از 8086

در این بخش می خواهیم مفاهیم اصلی مربوط به اینترفیس 8086 را به تراشه های جانبی مانند حافظه و I/O بررسی کنیم.

پایه ها و سیگنال های 8086

پایه ها و سیگنال های 8086 در شکل زیر نشان داده شده است.

سیگنالهای معمولی

نام	تابع	نوع
ADI5-AD0	باس آدرس / داده	دوطرفه
A19/56-A16/S3	آدرس/وضعیت	خروجی سه حالته
$\overline{BHE}$/S7	Status/Bus high enable	خروجی سه حالته
MN/$\overline{MX}$	Minimum/Maximum در کنترلی	ورودی
$\overline{RD}$	کنترل خواندن	خروجی سه حالته
TEST	انتظار برای کنترل تست	ورودی
READY	کنترل وضعیت انتظار	ورودی
RESET	Reset سیستم	ورودی
INTR	Interrupt request	ورودی
CLK	کلاک سیستم	ورودی

GND	زمین	ورودی

سیگنالهای مد می نیمم

نام	تابع	نوع
HOLD	Hold request	ورودی
HLDA	Hold ACK	خروجی
$\overline{WR}$	کنترل نوشتن	خروجی سه حالته
M/$\overline{IO}$	Memory/IO Control	خروجی سه حالته
DT/$\overline{R}$	Data Transmit/Recive	خروجی سه حالته
$\overline{DEN}$	Data Enable	خروجی سه حالته
ALE	Address Latch enable	خروجی سه حالته
INTA	Interrput ACK	خروجی

سیگنال مد ماکزیمم $(MN/\overline{MX} = GND)$

نام	تابع	نوع
$\overline{RQ/GT1,0}$	Hold request Access Control	دوطرفه
$\overline{LOCK}$	Bus priority lock Control	خروجی سه حالته
$\overline{S2}$, $\overline{S0}$, $\overline{S1}$	Bus cycle status	خروجی سه حالته
QS1, QS0	Instruction Queve Status	خروجی

تمامی پایه های 8086 از نوع TTL هستند. همانطوری که قبلاً گفتیم 8086 می تواند در دو مد مختلف عمل کند. می نیمم مد (تک پردازنده) و ماکزیمم مد (چند پردازنده ای). پایه ورودی $MN/\overline{MX}$ برای انتخاب یکی از این دو مد بکار می رود. اگر $MN/\overline{MX} = high$ باشد، 8086 در مد می نیمم کار می کند. در این مد، 8086 برای پشتیبانی می نیمم در سیستم های تک پردازنده ای که دارای وسایل کمی هستند که از باس استفاده می کنند، به کار می رود.

اگر $MN/\overline{MX} = low$ باشد، 8086 در مد ماکزیمم تعریف می شود تا از سیستم های چند پردازنده ای پشتیبانی کند. در این حالت اینتل 8288 که یک باس کنترلر است به 8086 برای فراهم ساختن کنترل های باس و سازگاری با معماری چندگذرگاهی اضافه می شود.

AD_0-AD_{15} خطوط گذرگاه ادغام شده آدرس/داده 16 بیتی هستند. در اولین سیکل کلاک AD_0-AD_{15} 16 بیت مرتبه پائین آدرس می باشند. 8086 آدرس 20 بیتی دارد و چهار خط مرتبه بالای ادرس، با سیگنال های وضعیت 8086 مالتی پلکس هستند. این سیگنال ها عبارتند از:

A16/S3 و A17/S4 و A18/S5 و A19/S6

در خلال اولین پریود کلاک از سیکل گذرگاه (سیکل نوشتن یا خواندن)، کل آدرس 20 بیتی در این خطوط در دسترس است. در تمامی دیگر سیکل های کلاک برای عملیات حافظه و I/O، ADI5-AD0 شامل داده 16 بیتی است و S3 و S4 و S5 و S6 خطوط وضعیت هستند.

خطوط S3 و S4 به شرح زیر دیکلود می شوند:

تابع	A16/S3	A17/S4
Extra Segment	0	0
Stack Segment	1	0
Code or no	0	1

Segement

1 1 Data Segment

بنابراین پس از اولین سیکل کلاک اجرای دستورالعمل، پایه های A17/S4 و A16/S3 مشخص می کنند که کدام ثبات سگمنت بخش سگمنت آدرس 8086 را تولید می کند. بدین ترتیب، با دیکود کردن این خطوط و استفاده از خروجی های دیکودر به عنوان انتخاب کننده های تراشه (Chip Selects) برای تراشه های حافظه، حداکثر تا 4 مگابایت (1MB در هر سگمنت) را می توان فراهم ساخت. این امر درجه ای از حفاظت را به وسیله اجتناب از عملیات نادرست نوشتن در یک سگمنت که با سگمنت دیگر همپوشانی دارد و نابود کردن اطلاعات آن سگمنت فراهم می آورد.

A18/S5 و A19/S6 به عنوان A18 و A19 در خلال اولین پریود کلاک از اجرای دستور، به کار میروند. چنانچه یک دستور I/O اجرا شود، در خلال اولین پریود کلاک low باقی می ماند. در خلال تمامی دیگر سیکل ها، A18/S5 وضعیت پرچم تواناساز وقفه 8086 را نشان می دهد و A19/S6 به S6 تبدیل می شود و Low بودن پایه A19/S6 نشان می دهد که 8086 در گذرگاه است. در اثنای Hold Acknowledge پریود کلاک، 8086 پایه A19/S6 را سه حالته کرده وبدین ترتیب به bus master دیگر برای گرفتن کنترل گذرگاه سیستم اجازه می دهد.

8086 ، AD0-AD15 را در خلال سیکل Interrupt ACK یا سیکل Hold ACK سه حالتی می کند.

$\overline{BHE}$/S7 به عنوان Bus high enable در خلال اولین سیکل کلاک اجرای دستور به کار میرود. 8086 در این پایه در هنگام خواندن، نوشتن و Interrupt Ack که داده ها به شکل مرتبه بالای گذرگاه داده (AD15-AD8) انتقال می یابند، low قرار میدهد. هم چنین BHE می تواند با اتصال با AD0 برای انتخاب بانک های حافظه استفاده شود. در خلال تمامی دیگر سیکل ها BHE/S7 به صورت S7 استفاده می شود و 8086 دارای سطح خروجی ($\overline{BHE}$) اولین سیکل در این پایه خواهد بود.

هنگامی که 8086 داده ها را از حافظه یا یک مکان I/O می خواند، $\overline{RD}$ = Low خواهد بود.

$\overline{TEST}$ یک پایه ورودی است و فقط به وسیله دستور WAIT استفاده یم شود. 8086 یک وضعیت Wait را پس از اجرای دستور WAIT تا هنگامی که Low در پایه TEST باشد قرار میدهد. این ورودی به شکل داخلی در اثنای هر سیکل کلاک در لبه کلاک سنکرون می شود.

INTR یک ورودی وقفه maskable است. این خط latch نمی شود و بنابراین تا هنگام تشخیص و تولید وقفه باید در high نگاه داشته شود.

NMI ورودی non maskable interrupt است. RESET سیگنال ورودی reset سیستم است و این سیگنال باید برای حداقل چهار سیکل تا هنگام تشخیص و شناسایی high باشد و سبب می شود رجیسترهای DS، SS، ES، Ip , و Flagها همگی صفر شوند.

هم چنین CS به FFFFH تنظیم می گردد. پس از حذف reset از پایه RESET، 8086 دستورالعمل بعدی را از آدرس فیزیکی 20 بیتی FFFF0H یعنی (CS:FFFFH,IP=0000) CS*16+IP واکشی می کند.

هنگامی که 8086 لبه هشت پالس را در RESET تنشخیص دهد، تمامی فعالیت های خویش را تا هنگام low شدن کلاک متوقف می سازد. هرگاه reset برابر low گردد، 8086 سیستم را به شرح زیر مقداردهی اولیه (initislize) می کند.

مؤلفه های 8086	محتوی
پرچم ها	Clear
IP	0000H
CS	FFFFH
DS	0000H
SS	0000H
ES	0000H
Queue	Empty

سیگنال reset برای 8086 را می توان با 8284 تولید کرد. 8284 یک ورودی تریگر اشمیت (Schmitt Trigger) به نام $\overline{RES}$ برای تولید reset از خروجی reset دارد.

برای تولید reset به شکل صحیح، ورودی باید برای 50MS پس از این که Vcc به می نیمم ولتاژ 4.5V رسید زیر 1.05 ولت نگاهداشته شود. سیگنال ورودی $\overline{RES}$ از 8284 را می توان با مدار RC ساده ای شبیه شکل زیر ساخت:

مقادیر R و C به شرح زیر قابل حصول هستند:

$$VC(t) = V(1 - e^{-t/RC})$$

بطوریکه

$RC = 188, Vc = 1.05V, V = 4.5V, t = 50s$ میکروثانیه

مثلاً اگر C را به شکل اختیاری $0.1 \mu F$ انتخاب کنیم $R=1.88K\Omega$

همانطوری که قبلاً گفتیم 8086 می تواند در مد مینیمم و ماکزیمم با استفاده از پایه $MN/\overline{MX}$ پیکره بندی شود. در مد می نیمم، 8086 بخودی خود تمامی سیگنال های کنترل گذرگاه را تولید می کند. این سیگنال ها عبارتند از:

- $DT/\overline{R}$ (Database Transmit/ReCeive)

یک سیگنال خروجی لازم برای مد 8086 می نیمم است که از 8286/8287 (data bus transceiver) استفاده می کند و برای کنترل جهت جریان داده ها استفاده می شود.

- (Data Enable) $\overline{DEN}$

به شکل یک output enable برای 8286/8287 در سیستم می نیمم استفاده می شود.

$\overline{DEN}$ در خلال هر دستیابی بحافظه و I/O برای سیکل های $\overline{INTA}$، برابر low می باشد.

- (Address Latch Enable) ALE

یک سیگنال خروجی فراهم شده توسط 8086 است و برای دی مالتی پلکس کردن AD0-AD15 به A0-A15 و D0-D15 در لبه پائین رونده ALE استفاده می شود. سیگنال ALE در 8086 مشابه ALE در 8085 است.

- $M/\overline{IO}$ این سیگنال خروجی برای تمایز دستیابی به حافظه ($M/\overline{IO}$=high) از ($M/\overline{IO}$=low) I/O استفاده می شود. هنگامی که 8086 یک دستور I/O را مانند IN یا OUT تولید می کند، این پایه را به low تنظیم می کند و هنگامی که رجوع به حافظه را مثل [SI] و MOV AX اجرا می کند، این پایه را به high تنظیم می کند.

- $\overline{WR}$

در هنگام نوشتن در حافظه یا I/O به low تنظیم می شود.

- $\overline{INTA}$ مانند $\overline{INTA}$ در 8085 است. و برای سیکل های تأئید وقفه این پایه low می شود.

- ورودی HOLD و خروجی HLDA

از این پایه ها در DMA استفاده می شود. High بودن این پایه یعنی این که یک master دیگری تقاضای در اختیار گرفتن سیستم Bus را دارد و پایه HLDA را به عنوان ACK

برابر high می کند و به طور همزمان پردازنده، گذرگاه را سا حالته کرده تا هنگامیکه برای پایه HOLD سیگنال low را دریافت کند. در این صورت پردازنده low را در پایه HLDA قرار میدهد. HOLD یک ورودی آسنکرون نیست.

- CLK

ورودی فراهم کننده زمان بندی اصلی برای 8086 و کنترل کننده گذرگاه

در مد ماکزیمم برخی از پایه های 8086 به صورت دوباره تعریف می شوند. مثلاً پایه های HOLD، HLDA، $\overline{WR}$، $M/\overline{IO}$، $DT/\overline{R}$، $\overline{DEN}$، ALE و $\overline{INTA}$ در مد می نیمم به شکل $\overline{RQ/GT0}$، $\overline{RQ/GT1}$، $\overline{LOCK}$، $\overline{S2}$، $\overline{S1}$، $\overline{S0}$، QS0 و QS1 به ترتیب در مد ماکزیمم تعریف می شوند. درمد ماکزیمم کنتلر 8288 اطلاعات وضعیت را از $\overline{S0}$، $\overline{S1}$، $\overline{S2}$ برای تولید زمان بندی گذرگاه و سیگنال های کنترل لازم برای سیکل گذرگاه دیکو.د می کند. $\overline{S0}$، $\overline{S1}$، $\overline{S2}$ در خروجی 8086 به شرح زیر دیکود می شوند:

توضیح	$\overline{S0}$	$\overline{S1}$	$\overline{S2}$
int. ACK	0	0	0
خواندن پورت I/O	1	0	0
نوشتن در پورت I/O	0	1	0
halt	1	1	0

دستیابی به کد	0	0	1
خواندن حافظه	1	0	1
نوشتن در حافظه	0	1	1
غیرفعال	1	1	1

- $\overline{RQ}/\overline{GT0}$ و $\overline{RQ}/\overline{GT1}$

این پایه ها به معنای request/grant و به وسیله bus master های محلی دیگر برای رهاسازی گذرگاه محلی توسط پردازنده در پایان سیکل گذرگاه جاری استفاده می شوند.

$\overline{RQ}/\overline{GT0}$ و $\overline{RQ}/\overline{GT1}$ دارای اولویت بیشتری از $\overline{RQ}/\overline{GT1}$ و این پایه به شرح زری عمل می کنند.

۱- یک پالس (به پهنای یک کلاک) از busmaster محلی دیگر نشان دهنده یک درخواست برای گذرگاه محلی به 8086 است.

۲- در پایان سیکل گذرگاه جاری یک پالس از 8086 به Master درخواست کننده نشان می دهد که 8086 سیستم باس را رها کرده و خروجی سه حالته است. بنابراین، bus master جدید با ارسال low در $\overline{RQ}/\overline{GT0}$ یا $\overline{RQ}/\overline{GT1}$ کنترل را واگذار می کند و 8086 آن را دوباره به دست می گیرد.

- $\overline{LOCK}$

8086 مقدار low را در پایه $\overline{LOCK}$ برای ممانعت از بدست آوردن کنترل باس سیستم توسط bus master دیگر قرار میدهد.

- QS0 و QS1

8086 برای فراهم ساختن بررسی خارجی وضعیت صف دستورالعمل 8086 به شرح زیر از QS0 و QS1 استفاده می کند:

	QS0	QS1
No operation	0	0
اولین بایت opcodo از صف	1	0
خالی بودن از صف	۰	۱
بایت بعدی صف	1	1

QS0 و QS1 در خلال پریود کلاکی که به دنبال هر عمل صف می آید، معتبر هستند. 8086 می تواند از 5V+ تا 10V+ عمل کند. برای کاهش نویز منبع تغذیه از دو GND در سیستم استفاده شده است.

سیکل گذرگاه 8086

به منظور ارتباط با دستگاه های خارجی از طریق سیستم گذرگاه برای انتقال داده ها یا واکشی دستورالعمل ها، 8086 یک سیکل گذرگاه را اجرا می کند. نمودار زمان بندی سیکل اصلی

گذرگاه 8086 در شکل زیر نشان داده شده است. می نیمم سیکل گذرگاه شامل چهار پریود کلاک cpu است که T sate نام دارند.

۱- در اولین T sste (T1)، پردازنده 8086 آدرس 20 بیتی محاسبه شده از زوج سگمنت/افت را در باس مالتی پلکس شده آدرس/داده/وضعیت قرار می دهد.

۲- برای دومین وضعیت T (T2)، 8086 آدرس را از روی باس برداشته و خطوط $AD15-AD_0$ را فعال یا سه حالته می کند تا برای خواندن داده از طریق خطوط $AD15-AD_0$ در خلال T3 آماده شوند.

در سیکل نوشتن در گذرگاه، 8086 داده ها را در خطوط $AD15-AD_0$ قرار می دهد. هم چنین در خلال T_2، چهار مرتبه خط بالای باس که مالتی پلکس شده اند از آدرس A19-A16 به وضعیت سیکل گذرگاه یعنی S6، S5، S4 .و S3 سوئیچ می کنند. 8086 سیگنال $\overline{RD}$ (سیکل خواندن) را low کرده و یا اگر بخواهد سیکل نوشتن را انجام دهد در خلال T2، T3 و T4 سیگنال WR را load می کند.

۳- در خلال T3، 8086 به قرار دادن اطلاعات وضعیت در چهار خط A19-A16/S6-S3 مبادرت می ورزد و داده ها را از خطوط $AD15-AD_0$ میخواند یا در آن خطوط می نویسد.

چنانچه وسیله I/O با حافظه مورد نظر بقدر کافی برای انتقال داده ها به 8086 کافی نباشد، خط ورودی READY در شروع T3 در وضعیت low قرار می گیرد و 8086 سیکل های کلاک

اضافی (وضعیت های انتظار TW) را پس T3 از درج می کند. فعالیت گذرگاه در خلال TW شبیه T3 است. در صورتیکه وسیله مورد نظر زمان کافی برای تکمیل انتقال داده ها داشته باشد، باید سیگنال پایه READY را high کند. به مجرد این که پریودهای کلاک TW تمام می شود، 8086 آخرین سیکل گذرگاه را اجرا می کند (T4). 8086 داده ها را در خطوط $AD15-AD_0$ در خلال آخرین وضعیت انتظار یا در اثنای T3 در صورتیکه حالت انتظاری نیاز نباشد، لچ خواهد کرد.

۴- در خلال T4، 8086 خطوط زمان را disable کرده و هم چنین حافظه و I/O انتخاب شده از باس را از کار می اندازد. یعنی سیکل گذرگاه در T4 پایان می پذیرد.

۵- پایه های $DT/\overline{R}$ $\overline{DEN}$ توسط 8286/8287 در می نیمم سیستم استفاده می شوند. در خلال سیکل خواندن 8086، DEN را در خلال بخشی از T2 و تمامی سیکل T3 برابر low قرار می دهد. این سیگنال می تواند برای enable کردن 8286/8287 استفاده شود. از شروع T1 و بخشی از T4 پایه $DT/\overline{R}$ برابر low می شود و 8086 از این سیگنال برای خواندن یا دریافت داده ها از گیرنده در خلال T3 تا T4 استفاده می کند.

در اثنای سیکل نوشتن، $\overline{DEN}$ را برابر low قرار می دهد (بخشی از T1 و تمام T2 و T3) و بخشی از سیکل T4. این سیگنال می تواند برای enable کردن Transcerver به کار آید. در چهار

سیکل گذرگاه در DT/R مقدار high را برای نوشتن یا ارسال داده ها به Transcerver در خلال T3 تا T4 قرار می دهد.

مفاهیم گذرگاه داده و گذرگاه آدرس در 8086

مهمترین قابلیت اینترفیس کردن تراشه های حافظه و I/O به 8086 نیاز به آدرس پایداری برای مدت زمان سیکل گذرگاه است. به همین خاطر، آدرس در گذرگاه مالتی پلکس شده آدرس / داده در خلال T1 باید latch شود. آدرس latch شده سپس برای انتخاب I/O یا خانه حافظه مورد نظر استفاده می شود. دقت کنید 8086 دارای باس آدرس و داده مالتی پلکس شده 16 بیتی است. بنابراین، برای دی مالتی پلکس کردن گذرگاه، 8086 از سیگنال ALE برای در اختیار گرفتن آدرس در لچ های 8 بیتی 8282 یا 8283 استفاده می کند. این لچ ها تا مادامیکه ALE=high است، آدرس را به خروجی منتقل می کنند و در لبه پائین رونده ALE آدرس را لچ می کنند. شکل زیر نشان میدهد که چگونه 8086 با سهای آدرس و داده را دی مالتی پلکس می کند.

معمولاً برنامه نویس حافظه 8086 را فضای آدرس شامل یک میلیون بایت متوالی داده ای می بیند که محتوی هر خانه 8 بیت و دو خانه 16 بیت می باشد. در آدرس های بایت یا کلمه محدودتی

وجود ندارد. اما در واقع فضای آدرس به شکل فیزیکی در گذرگاه داده 16 بیتی بنا شده است و فضای آدرس به دو بانک 512 کیلو.بایتی تقسیم می شود.

این بانک ها می توانند توسط $\overline{BHE}$ و A0 به شرح زیر انتخاب شوند:

بایت انتقال یافته	A0	$\overline{BHE}$
هر دو بایت	0	0
بایت مرتبه بالا به / از آدرس فرد	1	0
بایت مرتبه پائین به / از آدرس زوج	0	1
هیچ کدام	1	1

یک بانک به D7-D0 متصل است و شامل تمامی بایت هایی با آدرس زوج (A_0=0) است. بانک دیگر به D15-D8 متصل است و شامل تمامی بایت های با آدرس فرد (A_0=1) است. هر بایت در هر بانک را می توان با A19-A1 آدرس دهی کرد. A0=low سبب فعال شدن بانک با آدرس زوج می شود و بایت های داده در خطوط D7-D0 انتقال می یابند. 8086 سیگنال high را در $\overline{BHE}$ قرار داده و بانک آدرس های فرد را disable می کند و هم چنین با low کردن این

پایه و high کردن A0 می تواند بانک شامل آدرس‌های زوج را اختیار کند. این مسأله سبب انتقال داده ها به نیمی از باس داده خواهد شد. فعال ساختن A0 و BHE توسط 8086 انجام می گیرد و به آدرس های زوج یا فرد بستگی دارد و برای برنامه نویس شفاف می باشد.

مثال: دستور [BX] و DH MOV را در نظر بگیرید.

چنانچه آدرس 20 بیتی توسط DX و DS زوج باشد، 8086 مقدار A0 را low کرده و $\overline{BHE}$ را برابر high قرار می دهد. یعنی بانک حاوی آدرس های زوج انتخاب می شود. محتوای حافظه انتخاب شده در D7-D0 قرار می گیرد. 8086 این داده ها را خوانده اتوماتیک در DH قرار می‌دهد. اکنون دستیابی به کلمه 16 بیتی را توسط 8086 با آدرس زوج و بایت مرتبه کمتر در شکل زیر درنظر بگیرید.

مثلاً فرض کنید 8086 دستور CX و [BX] Mov را اجرا می کند. فرض کنید:

[BX] = 0004H

[DS]=2000H

آدرس 20 بیتی =20000+0004=20004H

8086 هم A0 و هم $\overline{BHE}$ را low می کند یعنی هر دو بانک فعال خواهند شد. محتوی [CL] به D7-D0 و [CH] به D15-D8 انتقال پیدا می کند. ($\overline{WR}$=Low و $M/\overline{IO}$=high) بانک های

حافظه شامل داده 16 بیتی برای نوشتن هستند و [CL] به مکان 20004H و [CH] در مکان 20005H نوشته خواهند شد.

اکنون فرض کنید که آدرس محاسبه شده توسط 8086 فرد باشد. مثلاً آدرس فیزیکی 20 بیتی برابر 20005H باشد. 8086 این کار را در دو سیکل گذرگاه انجام خواهد داد. در سیکل اول، 8086 باید A0=high و $\overline{BHE}$=low را انجام دهد. یعنی بانک محتوی آدرس فرد فعال و بانک محتوی آدرس زوج غیرفعال شود. هم چنین $\overline{RD}$=low و M/$\overline{IO}$=high.

در این سیکل گذرگاه، مکان [20004H] در خطوط D15-D8 قرار می گیرد. 8086 داده ها را از [CL] میخواند. در سیکل دوم A0 را Low کرده و $\overline{BHE}$را high می کنند و $\overline{RD}$=low و M/$\overline{IO}$=high می شود و بانک آدرس زوج انتخاب شده و [20006H] در خطوط D7-D0 قرار داده می شود. 8086 داده [CH] را می خواند. به شکل های زیر نگاه کنید:

در خلال خواندن یک بایت، 8086 کل خطوط $\overline{D15-D5}$را شناور می کند. این مسئله دیکود کردن انتخاب تراشه را برای ROM ها و EPROMها ساده می کند. در خلال نوشتن یک بایت، 8086 کل باس داده 16 بیتی را درایو خواهد کرد. اطلاعات در نیمه گذرگاه داده داده های معینی را انتقال نمیدهد.

چنانچه وسایل حافظه یا I/O مستقیماً به گذرگاه مالتی پلکس شده متصل شدند، طراح باید ضمانت کند که وسایل آدرس را در گذرگاه هنگام T1 خراب نخواهد کرد. برای اجتناب از این

مشکل، وسایل حافظه یا I/O باید دارای یک enable خروجی باشند که توسط سیگنال read پردازنده 8086 کنترل می شود. این مساله در شکل زیر نشان داده شده است.

اینترفیس کردن با حافظه:

ROM و EPROM

ROM ها و EPROM ها ساده ترین تراشه های حافظه برای اینترفیس با 8086 هستند. از آنجائیکه ROM و EPROM وسایلی read only هستند، لازم نیست که A0 و $\overline{BHE}$ به عنوان بخشی از دیکودینگ enable/select تراشه گنجانده شوند زیرا hip enable مشابه chip select است با این فرق که chip enable هم چنین مشخص می کند که آیا chip فعال است یا در حالت standby میباشد.

خطوط آدرس 8086 باید به تراشه های ROM/EPROM که با A1 شروع می شوند و بالاتر از آن که به تمامی خطوط تراشه های ROM و EPROM متصل هستند خطوط استفاده نشده آدرس 8086 می تواند برای دیکودینگ enable/select تراشه استفاده شود. برای اینترفیس

کردن ROM/RAM ها به گذرگاه مالتی پلکس شده 8086، آن ها باید دارای سیگنال های enable خروجی باشند. شکل بعدی اینترفیس 8086 را با دو ماجول 2716 نشان میدهد.

دستیابی به بایت توسط خواندن کامل 16 بیت کلمه در باس با کنار گذاشتن بایت غیر مطلوب و پذیرش بایت مورد نظر انجام می گیرد.

سیستم وقفه 8086

وقفه ها در 8086 را می توان به سه نوع زیر تقسیم بندی کرد:

۱- وقفه های از پیش تعریف شده (Predefind interrupts)

۲- وقفه های نرم افزاری تعریف شده توسط کاربر (User defined software interupts)

۳- وقفه های سخت افزاری تعریف شده توسط کاربر (User defined hardware interupts)

آدرس های بردار وقفه برای تمامی وقفه های 8086 از جدولی که در آدرس 00000H⇒003FFH ذخیره شده معین می شوند. آدرس شروع برای روتین های سرویس وقفه ها با استفاده از این جدول فراهم می شود. چهار بایت از جدول برای هر یک از از وقفه ها در نظر گرفته شده است. 2 بایت برای IP و 2 بایت برای CS. جدول می تواند تا 256 بردار 8 بیتی را شامل باشد. چنانچه کمتر از 256 وقفه در سیستم تعریف شده باشد، کاربر فقط نیاز به فراهم ساختن حافظه کافی برای اشاره گر به جدول وقفه برای فراهم ساختن وقفه های تعریف شده دارد.

برای آدرس وقفه (CS , IP) برای تمامی وقفه های 8086 به هر وقفه یک نوع کد را برای شناسائی وقفه اختصاص میدهد. 256 کد نوع وجود دارد و هر مدخل جدول شامل دو آدرس یکی برای ذخیره محتوی IP و دیگری برای ذخیره سازی محتوی CS است. هر بردار آدرس فیزیکی وقفه 8086 دارای 20 بیت است و از محتویات 16 بیتی IP و CS محاسبه می شود.

برای فراهم ساختن برادار آدرس وقفه، 8086 دو آدرس را در جدول اشاره گر جائیکه IP و CS برای نوع خاصی از وقفه ذخیره شده اند، محاسبه می کند. بطور مثال، برای وقفه نوع int nn داریم:

$$IP = 4 * nn$$

$$CS = 4 * nn + 2$$

برای سرویس دادن به وقفه NM1 در 8086، پردازنده 8086 کد نوع 2 را به این وقفه اختصاص داده است. 8086 به شکل اتوماتیک دستور INT2 را به شکل داخلی برای فراهم ساختن بردار آدرس وقفه به شرح زیر اجرا کند:

$$\text{آدرس برای IP} = 4 * 2 = 00008H$$

$$\text{آدرس برای CS} = 4 * 2 + 2 = 0000AH$$

8086 مقادیر IP و CS را از آدرس فیزیکی 20 بیتی 00008H و 0000AH در جدول اشاره گر load می کند. کاربر باید مقادیر 16 بیتی مناسب IP و CS را در این مکان ها قرار دهد. مشابهاً، مقادیر CS و IP برای دیگر وقفه ها قابل ملاحظه هستند. جدو.ل اشاره گر وقفه در 8086 در شکل زیر بچشم می خورد.

در پاسخ به یک وقفه، 8086، پرچم ها، CS و IP را در Stack دخیره (push) می کند و پرچم های TF و IF را clear کرده و سپس IP و CS را از جدول اشاره گر مربوط به کد نوع وقفه load می کند.

روتین های سرویس وقفه را باید با IRET (Interrupt Return) خاتمه داد که این دستور سه کلمه ای بالای stack یعنی IP، CS و پرچم ها را pop کرده و به مکانهای اصلی در برنامه اولیه بازگشت پیدا می کند.

256 کد نوع به صورت زیر مقداردهی می شوند:

- نوع 0 تا 4 برای وقفه های از پیش تعریف شده.
- نوع 5 تا 31 برای کارها و هدفهای آتی توسط اینتل رزرو شده اند.
- نوع 32 تا 255 برای وقفه های maskable در دسترس می باشند.

وقفه های از پیش تعیین شده (0 تا 4)

وقفه های ازپیش تعیین‌شده شامل division by zero (نوع 0)، Single step (نوع 1)، پایه NMI (نوع 2)، Break point Interupt (نوع 3) و interrupt on overflow (نوع 4) هستند. کاربر باید مقادیر مناسب IP و CS را از جدول وقفه فراهم سازد. چنانچه از وقفه پیش تعریف شده در سیستم استفاده نشود، کاربر می تواند توابع دیگری را به نوع مربوطه اختصاص دهد.

هرزمان که تقسیم بر صفر انجام شود، 8086 به شکل خودکار وقفه داده می شود. این وقفه non maskable است و توسط اینتل به عنوان بخشی از دستور تقسیم پیاده سازی است. هنگامی که

TF (trap frag) توسط دستوری set می شود، 8086 به مد single step میرود. TF را می توان به شکل زیر clear کرد:

Push F	Save flags ;
MOV BP,SP	[SP] [BP] ;
AND [BP+0] و 0FEFFH	; Clear TF
POP F	; POP FLAGS

در قطعه کد بالا، به جای [BP] از [BP+0] استفاده شده زیرا نمی توان بدون displacement استفاده کرد. اکنون برای set کردن TF به جای دستور AND می توان از دستور زیر استفاده کرد:

OR [BP+0] , 0100H

هر زمان که TF را به 1 تنظیم می کنیم، 8086 به شکل خودکار وقفه نوع 1 را پس از اجرای هر دستور تولید می کند. کاربر می تواند یک روتین سرویس را در بردار آدرس وقفه برای نمایش محتویات حافظه یا ثبات برای debug برنامه بنویسد. دقت کنید single step یک وقفه non maskable است و نمی تواند به وسیله STI (تواناساز وقفه) و یا CLI (ناتوان ساز وقفه) تغییر پیدا کند. وقفه های non maskable توسط پین NMI تنظیم می شود.

این وقفه edge trigger است (high → low) و باید برای دو سیکل کلاک (برای تخمین شناسایی) فعال شود و بطور نرمال برای خطاهای catastrophic مثل power failure استفاده می شود. 8086 آدرس بردار وقفه را به شکل خودکار با اجرای دستور int 2 فراهم می سازد.

Type 3 برای break point و non maskable استفاده می شود. کاربر دستور یک بایتی int 3 را در برنامه با جایگزین ساختن یک دستور درج می کند. Breaking برای debug کردن برنامه سودمند می باشد.

Type 4 وقفه برای سرریزی است و در صورتیکه فلگ OF تنظیم می شود و دستور into اجرا گردد اتفاق می افتد. مثلاً فلگ سرریزی پس از اجرای دستور ریاضی مثل MULS (signal multiplication) اتفاق می افتد. کاربر می تواند دستور into را پس از muls اجرا کند. چنانچه سرریزی وجود داشته باشد، روتین سرویس نوشته شده توسط کاربر که در آدرس وقفه نوع 4 نوشته شده، اجرا می گردد.

User defined software interrupts

کاربر می تواند با اجرای یک دستور وقفه 2 بایتی به نام int nn، وقفه نرم افزاری را تولید کند. این دستور با فلگ وقفه IF قابل mask شدن نمی باشد. هم چنین دستور int nn می تواند برای آزمون کردن روتین سرویس وقفه برای وقفه های خارجی به کار رود. کدهای نوع 0 تا 255 قابل استفاده هستند. چنانچه وقفه از پیش تعریف شده ای در سیستم استفاده نشود، کد نوع مرتبط با دستور int un را می توان برای تولید وقفه های داخلی یا نرم افزاری استفاده کرد.

شکل زیر تقسیم بندی انواع وقفه ها را را نشان می دهد:

مثال: آدرس فیزیکی و منطقی بردار وقفه های زیر را پیدا کنید:

۱- int 14H

۲- int 38H

1- IP ← 14*4=50H : CS ← 50H+2=52H

(محدوده آدرس) 0000:0050 0000:0053

2- IP ← 38*4=E0H: ← CS E0H+2=E2H

(محدوده آدرس) 0000:00E0 0000:00E3

مثال: فرض کنید محتوای حافظه و آدرس های متناظر با آن به صورت زیر باشند.

0000:0000 E8 56 2B 02 56 07 70 00-C3 E2

00 F0 56 07 70 00

0000:0010 56 07 70 00 54 FF 00 F0 -47 FF

00 F0 47 FF 00 F0

محتوی IP و CS را برای روتین سرویس وقفه int 5H پیدا کنید.

IP=5*4=0014H

CS=0014 + 2 =0016H

طبق فرض مسأله [IP]=FF54H

[CS]=F000H

user defined hardware (maskable interrupts)

وقفه های قابل mask در 8086 از طریق پایه intr مقداردهی اولیه می شوند. این وقفه ها می توانند به وسیلع دستور STI (1 ← (set If or IF یا ازطریق‌دستور CLI (0 ← IF یا clear IF) توانا و ناتوان شوند. چنانچه IF=1 و intr فعال (high) باشد، بـدون وجـود هرگونه وقفه دیگری، پردازنده 8086 پس از تکمیل دستور جاری دوبار $\overline{\text{int A}} = \text{low}$ را اجرا مـی کند. (هر بار در حدود ۲ سیکل)

وضعیت پایه Intr در طی آخرین سیکل ساعت هر دستور، نمونه برداری مـی شـود. در برخـی اوقات، 8086 پایه INTR را در زمانی دیگر نمونه گیری می کند. مثالی از این دست اجـرای دستور pop برای سگمنت رجیستر است. در این حالت، وقفه ها تا هنگام تکمیل دستور بعـدی نمونه برداری می شوند. این امر سبب می شود که 32 بیت اشاره گر بتواند در SS و sp بـدون مشکل رخداد وقفه در بین دو load، بارگذاری شود.

$\overline{\text{INTA}}$ توسط 8086 در پاسخ به INTR تولید می شود. به شکل زیر نگاه کنید.

ترتیب تأئید وقفه شامل دو سیکل $\overline{INTA}$ است که با دو سیکل کلاک idle از یکدیگر جدا می شوند. ALE نیز بوسیله 8086 تولید می شود و آدرس لچ شده را در باس load می کند. در طی سیکل های باس $\overline{INTA}$، DT/$\overline{R}$ و $\overline{DEN}$ low هستند. یعنی در نخستین سیکل باس $\overline{INTA}$ نشان میدهد که یک سیکل تأئید وقفه در حال پیشرفت است و به سیستم اجاره می دهد برای قرار دادن کد نوع وقفه در سیکل باس $\overline{INTA}$ بعدی آماده میباشد. پس 8086 نمی تواند اطلاعات را در طی اولین سیکل باس فراهم کند. سخت افزار خارجی باید کد نوع را در نیمهٔ پائین باس داده 16 بیتی در طی سیکل دوم قرار دهد.

8086 DMA

هنگامی که در مد می نیمم هستیم، 8086 سیگنال HOLD (درخواست DMA) و HLDA (تأئید DMA) را برای در اختیار گرفتن باس برای کاربردهای DMA تولید می کند. تراشه های کنترلر DMA یعنی 8257 و 8237 می توانند با 8086 بکار روند. این تراشه ها می توانند با فعال کردن پایه HOLD درخواست انتقال DMA را بین حافظه 8086 و وسیله I/O انجام دهند. 8086 سیکل گذرگاه جاری را کامل می کند و سپس HLDA را فعال کرده و باس سیستم را به کنترلر DMA واگذار می کند. 8086 تا هنگامیکه پایه HOLD منفی نشود سعی در استفاده کردن از آن نمی کند.

همانطوری که قبلاً گفتیم آدرس های حافظه 8086 در دو بانک جداگانه که یکی شامل آدرس های زوج و دیگری فرد است پیکره بندی می شوند. کنترلر DMA هشت بیتی متناوباً این دو بانک را برای دستیابی به بایت های منطقی مجاور در حافظه انتخاب کند.

- **فلسفه استفاده از DMA**

بسیاری از وسایل I/O برروی بلوک های بزرگ از داده ها عمل می کنند (معمولاً چندین کیلوبایت طول دارند). این مسئله سبب می شود که تاخیر عملیات طولانی مثل دستیابی به هارددیسک در کل انتقال بلوک کاهش پیدا کند. این وسایل معمولاً دارای یک بافر کوچک حافظه هستند که بلوک هایی از داده را در خود نگاه مید ارد. ثابت های فرمان نگاشت حافظه وسیله سبب می شوند که پردازنده بتواند کلمات را از/به بافر با عملیات load و store بخواند یا بنویسد.

با چنین طرحی پردازنده باید عملیات load یا store را برای هر کلمه از داده ها کنید به یا از بافر وارد/خارج می شوند، اجرا کند. درحالیکه تعاملات روی باس I/O معمولاً در سرعتی کمتر از سرعت کلاک پردازنده روی میدهند. یعنی وقت کافی بین نقل و انتقالات گذرگاه

I/O برای پردازنده باقی نمیماند تا به برخی از دیگر taskها سوئیچ کند. یعنی به زبان ساده تر پردازنده باید هر زمان که داده ها روی گذرگاه I/O رد و بدل می شوند مشغول و منتظر باشد.

سیستم های DMA برای حل این مشکل بوجود آمده اند. در سیستم DMA، وسایل I/O می توانند مستقیماً و بدون دخالت پردازنده به حافظه دستیابی پیدا کنند. شکل زیر مفهوم DMA را نشان میدهد.

بکمک DMA می توان تعداد سیکل هایی را که یک ریزپردازنده صرف I/O می کند کاهش داد و درنتیجه پردازنده برای انجام کارهای دیگر آزاد خواهد بود. هرچند که، وسیله I/O و پردازنده مجبور به استفاده از پهنای باند مشترک حافظه هستند یعنی این که پهنای باند موجود حافظه برای برنامه ها در هنگام استفاده از DMA کاهش پیدا خواهد کرد.

ساختار عمومی ریزپردازنده های پیشرفته

در این بخش ساختار ریزپردازنده های اواسط دهه ۹۰ شرح داده می شوند. این ساختار در بیشتر ریزپردازنده های این دوره وجود دارد و به ریزپردازنده های خاصی مربوط نیست. هرچند چندین سیستم 64 بیتی در این دوره وجود دارد، بیشتر ریزپردازنده های این دوره 32 بیتی هستند. شکل زیر نمودار بلوکی ریزپردازنده ها را ناشن می دهد:

تعداد سیستم های 64 بیتی روز به روز افزایش می یابند و حتی در سیستم های 32 بیتی نیز تعداد گذرگاههای داخلی و خارجی می توانند چندین برابر باشند. در سیستم های 32 بیتی، Iu و ثبات های موجود در فایل ثبات 32 بیتی هستند و گذرگاه داده ممکن است 32 بیتی باشد. البته سیستم هایی نیز وجود دارند که در آن ها گذرگاه داده ممکن است 64=2*32،

96=3*32 یا 128=4*32 بیتی باشند. در سیستم های 64 بیتی Iu، و ثبات ها در فایل ثبات 64 بیتی هستند و گذرگاه ها نیز می توانند ضریبی از 64 باشند. گذرگاه آدرس در بیشتر سیستم ها 32 بیتی است که برای آدرس دهی فضای 4GB کافی می باشد. اگرچه برخی سیستم های جدید از این هم فراتر رفته اند مثلاً 32+j که j یک عدد صحیح است.

BIU واسطی بین واحدهای داخلی ریزپردازنده و سیستم های خارجی است. جزئیات بیشتر BIU در شکل زیر نشان داده شده است:

BIU شامل سه بخش است:

۱- واسط داده

۲- واسط آدرس

۳- واسط کنترل

بخش واسط داده رابطی بین گذرگاه داده سیستم و واحدهای داخلی ریزپردازنده است. بطور عمومی انتقال داده‌ها بین واسط داده و دیگر اجزای داخلی ریزپردازنده توسط گذرگاه خارجی انجام می گیرد.

بخش واسط آدرس، آدرس های تولید شده داخلی یک دستورالعمل یا یک عنصر داده را به بخش آدرس گذرگاه سیستم ارسال می کند. این آدرس توسط (Memory Managemet unit) Mmu ساخته می شود.

بخش واسط کنترل نیز وظیفه ارسال و دریافت تعدادی سیگنال های کنترل و وضعیت را از پردازنده به سیستم های خارجی و از آن ها به پردازنده برعهده دارد. اکثر خطوط کنترل در واسط کنترلی به واحد کنترل متصل هستند و واحد کنترل نیز به بقیه واحدهای پردازنده مرتبط است. خطو کنترل خروجی نشان دهنده وضعیت عملکرد بخش های مختلف ریزپردازنده، فعال شدن و فرمان هایی برای خواندن یا نوشتن، تمیز دادن بین دسترسی به انواع مختلف فضای آزاد آدرس دهی و پاسخ به وقفه ها و تقاضای درخواست گذرگاه و خیلی از موارد دیگر است.

خطوط کنترل ورودی نیز نشان دهنده حالت تجهیزات خارجی مثل آگاه ساختن ریزپردازنده که در حالتی که سیستم یا گذرگاه دچار خطا شود، درخواست سرویس دادن یک وقفه یا جواب یک تقاضا برای استفاده از گذرگاه و درخواست یک چرخه گذرگاه کامل شده، غیرفعال شدن حافظه نهان داخلی و بسیاری از موارد دیگر است.

واحد پیش کشی شامل یک مدار منطقی است که وظیفه پیش کشی دستورالعمل ها را از حافظه نهان دستورالعمل و قرار دادن آن ها در یک صف دستورالعمل FIFO بر عهده دارد. یک صف دستورالعمل ممکن است شامل 8 تا 32 بایت باشد.

دستورهایی که از این صف خارج می شوند وارد واحد کدگشایی خواهند شد. این واحد دستورها را کدگشایی کرده و سیگنال های کنترلی لازم را در هر حالت به واحد کنترل ارسال می کند.

بیشتر ریزپردازنده های امروزی دارای یک سوپر اسکالر هستند. یعنی بیشتر از یک دستورالعمل در هر زمان برای کدگشایی وارد می گردد.

برخی ریزپردازنده ها ممکن است علاوه بر Iu و Fpu دارای یک واحد عملیات ویژه Sfu (Special Function unit) نیز باشند. یک Sfu ممکن است شامل یکی از موارد زیر باشد (یا هر واحد دیگر بسته به کاربرد):

۱- واحد گرافیکی

۲- واحد پردازش سیگنال

۳- واحد پردازش تصویر

۴- پردازنده بردار یا ماتریس

هر ریزپردازنده ممکن است بیشتر از یک Sfu داشته باشد. فناوری امروزی اجازه می دهد تا بیش از 100 میلیون ترازیستور را در یک تراشه جای دهند و با این ظرفیت قادر خواهیم بود چندین واحد عملیاتی را در یک تراشه جای دهیم. Sfu در یک تراشه تاخیر انتقال را در انتقال اطلاعات بین Sfuها با واحدهای دیگر حداقل گرده و باعث سرعت بیشتر اجرای عملیات می شود.

حافظه پنهان (cache) حافظه ای با سرعت دسترسی بالا است که بین cpu و حاذفظه اصلی قرار دارد. وجود آن باعث بالا رفتن کارآیی خواهد شد و به cpu اجازه میدهد که به اطلاعات موجود در حافظه با سرعت بالاتر دسترسی داشته باشد. امروزه ریزپردازنده های پیشرفته، 32KB یا بیشتر حافظه نهان تراشه دارند. عملاً تمام ریزپردازنده های پیشرفته به شکل pipe line عمل می کنند. معمولاً از یک حافظه Dual cache در ریزپردازنده های پیشرفته استفاده می شود یک cache برای دستور و یک cache برای داده که در شکل زیر نشان داده شده است.

در بسیاری از ریزپردازنده ها، حافظه Cache به واحد عملیاتی از طریق یک گذرگاه داده عملیاتی به نام ODB (Operation Data Bus) متصل است. OBD به اندازه کافی بزرگ است (128 یا 256 بیت) طوری که می تواند تعدادی دستورالعمل را در یک زمان انتقال دهد و این سبب بالا رفتن سرعا عملیات ریزپردازنده خواهد شد.

حافظه Cache ثانویه بین حافظه نهان اولیه و حافظه اصلی در سلسله مراتب حافظه قرار می گیرد. زمان دسترسی از حافظه اصلی کمتر است و خارج از تراشه می باشد و می توانند خیلی بیشتر از حافظه نهان اولیه باشد. وجود Cache دوم کارآیی سیستم را در دسترسی سریع بحافظه افزایش می دهد.

داخل تعدادی از ریزپردازنده های پیشرفته یک نوع BTC (Branch Target Cache) وجود دارد. در سیتم پاپ لاین هنگامی که با دستورالعمل پرش مواجه شدیم، دستورهای زیر دستور پرش باید از خط لوله بیرون روند و دستورالعمل مقصد باید به خط لوله واکشی شود. اگر دستور از حافظه باشد تاخیر زیادی در اجرای دستورها باعث خواهد شد. چنانچه دستورهای مقصد به داخل یک BTC در تراشه ارسال شوند، می توانند سریعتر به خط لوله واکشی شوند. بدین ترتیب کارآیی پردازنده افزایش پیدا می کند. در برخی سیستم ها BTC فقط شامل آدرس های مقصد و نه دستورهای مقصد می باشند.

همانطور که قبلاً گفته شد بیشتر ریزپردازنده های پیشرفته دارای خاصیت اجرای موازی دستورالعمل ILP هستند. در بیشتر حالات از یک سوپر اسکالر استفاده می شود. در یک سیستم سوپر اسکالر I دستورالعمل توسط واحد پیش واکشی در یک زمان واکشی شده و به واحد کدگشایی ارسال می شوند. (به شکل زیر نگاه کنید).

معمولاً تعدا واحدهای عملیاتی خط لوله ای برابر تعداد حالات ILP نیست. اگر برای کارآیی بهتر، I دستور را در یک زمان صادر (issue) کنیم سبب می شود که I دستور در هر زمان بخوبی اجرا گردد. شکل فوق حالتی را نشان می دهد که تعداد واحدهای عملیاتی خط لوله ای برابر تعداد سطوح موازی I باشد که عمل این شرط همیشه امکانپذیر نیست. زیرا محدودیت هایی که مساحت های سیلیکانی ایجاد می کنند سبب می شود که نتوانیم تعداد کافی واحد عملیاتی را در یک تراشه قرار دهیم. امکان مخاطرات خط لوله (hazard) مثل وابستگی داده در دستور بعدی و دستورهای کنترلی (مانند پرش) ممکن است سبب stall و تاخیر در خط لوله شوند.

واحد کنترل هم می تواند به شکل hard wired و هم به صورت micro programmed پیاده سازی شوند. بیشتر مدل های سنتی ریزپردازنده های cisc واحد کنترل معمولاً ریزبرنامه ریزی دارند ولی مدل های ریزپردازنده های RISC دارای کنترل سخت افزاری پیاده سازی می شود و این مساله سبب بالا رفتن سرعت و یکی شدن چرخه اجرا در همه یا بیشتر دستورها خواهد شد.

جزئیات یک Iu در شکل زیر نشان داده شده است:

باید تعدادی واحد عملیاتی موازی در سیستم سوپراسکالر وجود داشته باشد. بنابراین ممکن است به تعدادی واحد جمع، تفریق و ضری و تقسیم در Iu نیاز داشته باشیم. در آن یک رجیستر به نام رجیستر فایل صحیح IRF یا Integer Register File قرار دارد که شامل N رجیستر 32 یا 64 بیتی (بسته به 32 یا 64 بیتی بودن سیستم) است.

کامپیوترهای CISC معمولاً 8 تا 16 ثبات دارند ولی ریزپردازنده های RISC معمولاً بیش از 32 ثبات دارند. تعدادی از سیستم های RISC بیش از 100 ثبات دارند. واحد توزیع کننده دستورالعمل (inst. dispatcher) دستورهای کدگشایی شده را از واحد کنترل گرفته و این دستورها را به واحد عملیاتی مربوطه ارسال می کند. دستورهای مربوط به اعداد صحیح به سوی Iu و دستورهای اعشاری به سوی Fpu ارسال می گردند و دستورهای عملیات ویژه به سوی Sfu میروند. داده های مورد نیاز واحد عملیاتی از طریق ODB به Cache ارسال می شوند. بیشتر سیستم ها یک شیفت شبکه ای دارند که برای انجام عمل شیفت چند بیتی در یک چرخه استفاده می شود.

مسیر داده Fpu در شکل زیر نشان داده شده است. جریان اطلاعات شبیه Iu می باشد. در این حالت یک فایل ثبات ممیز شناور در FRF (Floating point Register File) وج.د دارد که شامل دو ثبات می‌باشد. در سیتم های CISC تعداد ثبات های FRF معمولاً 8 عدد است ولی در سیستم های RISC معمولاً 32 عدد هستند.

عملاً تمام ریزپردازنده های مدرن از استاندارد IGGE-754-1985 استفاده می کنند که برای دقت ساده به شکل 32 بیتی و برای دقت مضاعف به شکل 64 بیتی می باشند. حتی اگر سیستم 32 بیتی باشد. در برخی سیستم ها FRF از ثبات های 64 بیتی استفاده می کند. البته برخی از سیستم ها از فرصت دیگری که 80 بیتی است استفاده می کنند (VAX) و بهمین جهت دراین سیستم‌ها FRF ممکن‌است ۸۰ بیتی باشد. DEC Alpha از استاندارد IEEE بخوبی فرصت توسعه‌یافته VAX (Extension Precision) استفاده می کند.

عملیات اصلی MMU بصورت زیر است:

۱- تبدیل آدرس مجازی (منطقی) به آدرس فیزیکی (حقیقی) و ارسال آدرس فیزیکی به حافظه نهان یا به Biu و گذرگاه آدرس برای تجهیزات خارجی

۲- انجام عملیات صفحه بندی با حافظه مجازی که از طریق واحد paging صورت میگیرد.

۳- فراهم ساختن انجام مکانیزم قطعه بندی (در صورت وجود) توسط واحد قطعه بندی

۴- انجام حفاظت حافظه که در واحدهای صفحه بندی و قطعه بندی انجام می شود.

۵- مدیریت یک TLB یا ATC (Address Translation Cache) با سرعت دسترسی بالا برای تعدیل شماره صفحه ای مجازی به فیزیکی

عملاً تمام ریزپردازنده های پیشرفته واحد صفحه بندی و یک TLB و یا ATC دارند. فقط خانواده X86 دارای واحد مربوط به قطعه بندی است. در حالتی که در TLB یک miss روی دهد، MMu دارای منطقی است تا دسترسی به فهرست صفحه خاص و جداول در حافظه اصلی را نظارت کند و TLB را با شماره صفحه miss شده بار کند.

ریزپردازنده های مدرن ممکن است دارای صدها پین باشند. این ها اجازه میدهد که بتوانیم چندین گذرگاه داده یا گذرگاه آدرس جدا از هم داشته باشیم و نیز واسط های جداگانه ای برای گذرگاه سیستم و حافظه Cache ثانویه و کنترل های متنوع و سیگنال های وضعیت داشته باشیم. همچنین چندین پین برای Vcc و GND در تمام اطراف تراشه وجود دارد. این پین ها اجازه میدهند که مسیرهای تغذیه و زمین داخل تراشه کوتاه تر باشند. که درنتیجه طراحی تراشه ساده تر و هزینه آن و نیز تاخیر کاهش پیدا می کند.

معنائی در زندگی نیست . معنا را باید خلق کرد

اگر آن را خلق کردی ، معنا را کشف کرده ای

ذهنت باید آن را به وجود آورد .

نه ، تخت سنگ نیست که در گوشه ای افتاده باشد

ترانه ای است که باید آن را سرود .شیيء ؛ نیست ،

ارزشی است که با تکیه بر هشیاری آن را بدست می آوری

نوشته اُشو ترجمه عبدالعلی براتی شورشی

با احترام ، تقدیم به والدین عزیزم

منابع و مـأخذ:

1- Microprocessors & Micro Computers

8086 & Z-80 , John Effenbeck

2- Advanced Computer Architecture , Hwang

3- Advanced Microprocessors , Daniel Tabak.