پاسخ تمرین سری اول

درس سیستم عامل

دکتر زرندی

پویا پارسا – ۹۲۳۱۰۰۵

DMA و ISA DMA

- *DMA Fundamentals on Various PC Platforms, from A. F. Harvey and Data Acquisition Division Staff NATIONAL INSTRUMENTS*
- *http://wiki.osdev.org/ISA_DMA*
- *ece.ubc.ca, ELEC 379: DESIGN OF DIGITAL AND MICROCOMPUTER SYSTEMS 1998/99 WINTER*

در پردازش های کامپیوتری که نیاز به پردازش داده دارند، اطلاعات دستگاه های ورودی و خروجی اغلب نیاز است با سرعت بالا در حجم های بالایی مدیریت و پردازش شود. سه مکانیسم اصلی برای انتقال اطلاعات عبارتند از : Polling، وقفه ها (programmed IO) و دسترسی مستقیم به حافظه (DMA).

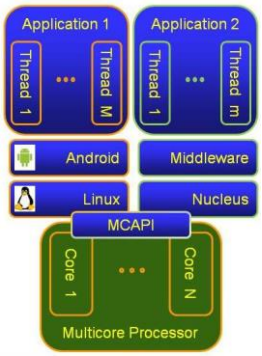
- در روش های **Polling** مثلPIO که یک عمل زمان بر است، برنامه با فراخواندن یک ساب روتین منتظر دریافت داده توسط سیستم عامل می ماند (این عمل اغلب توسط یک حلقه انتظار انجام می شود)
- در روش **Interrupt** پردازنده سیستم عامل به طور منظم در حین اجرای برنامه توسط وقفه داده های مورد نیاز را در یک بافر ذخیره می کند تا بعدا جهت پردازش استفاده شوند. این وقفه ها یک عملیات پیش زمینه هستند و برنامه نیاز به هیچ کدی برای خواندن از دستگاه ورودی ندارد. به جای آن سیستم عامل به طور نامرئی به اصطلاح در بازه های زمانی "دزدی" از نرم افزار این عمل را انجام می دهد.
- در روش **DMA** مثل PCIیک دستگاه تخصیص داده شده مستقیما داده ها را از ورودی خوانده و در حافظه ی بافر جهت استفاده ی پردازنده قرار می دهد. کاری که دستگاه **DMA** انجام می دهد از دید پردازنده عملا غیر قابل مشاهده است.
ایده ی اصلی که پشت DMA هست، می توان با ایجاد یک "کانال" با اختصاص دادن یک اشاره گر حافظه و طول مشخص داده برای انتقال ایجاد کرد. پردازنده می تواند به دستگاه خارجی اعلان کند که یک کار خاص مثلا خواندن یک سکتور را در آن کانال انجام دهد. هنگامی که باس حافظه توسط پردازنده مشغول نیست، چیپست دستگاه های خارجی بدون مشغول کردن پردازنده عملیات انتقال اطلاعات را انجام می دهد. بعد از اتمام کار و یا منتقل کردن یک بلاک به پردازنده سیگنال مناسب را ارسال می کند.

ISA DMA یا Industry Standard Architecture Direct Memory Access یک افزونه به PC های مدرن است که در کنترلر فلاپی دیسک ها، کارت های صدا، کارت های شبکه، و پورت های پارالل استفاده شده.مدار های کی بورد، تایمر و موس مدار های واضح و کاربرد های مشابهی دارند و استانداری که برای DMA آن ها از سال ۱۹۷۰ تعیین شده هنوز قابل کاربردند.

SMP و AMP

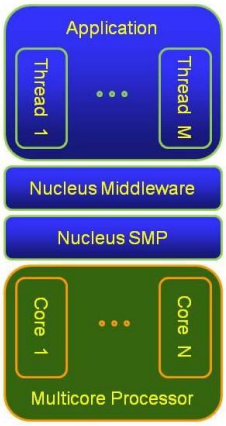
- *Lina J. Karam, Ismail AlKamal, Alan Gatherer, Gene A. Frantz, David V. Anderson, Brian L. Evans (2009). "Trends in Multi-core DSP Platforms" (PDF). IEEE Signal Processing Magazine, Special Issue on Signal Processing on Platforms with Multiple Cores.*
- *Mentor Graphics Blog, AMP vs SMP*

یکی از متداول ترین نکاتی که در پیاده سازی سخت افزار های Embedded رایج شده، بهره گرفتن از بیش از یک پردازنده است. شاید چند چیپ بر روی یک برد یا شاید توسط چند هسته در یک چیپ پردازنده. در روش های استفاده از چند هسته برای پردازش معماری هایی از جمله SMP و AMP وجود دارند.



ویژگی های **AMP** :

- چندین پردازنده
- هر کدام می توانند معماری متفاوت داشته باشند
- هرکدام فضای آدرس دهی و حافظه ی اصلی خودشان را دارند
- هر کدام می توانند سیستم عامل خودشان را داشته باشند و نداشته باشند
- یک روش ارتباطی مابین پردازنده ها وجود دارد



ویژگی های **SMP**:

- چندین پردازنده
- همه با یک معماری یکسان هستند
- به طور معمول یک سیستم عامل واحد تمامی پردازنده ها را مدیریت می کند و وظایف را بین آنها تقسیم می کند.
- یک روش اشتراکی جهت هماهنگ کردن پردازنده ها در اختیار قرار داده شده

SMP یا پردازش موازی موازن، به طور خلاصه یک معماری کامپیوتری است که با فراهم کردن امکان اجرای همزمان چند برنامه به طور موازی توسط چند پردازنده را بدون تاثیر منفی (Latency) بر روی عملیات را فراهم می کند. برخلاف معماری نامتقارن هر هسته ی بیکار می تواند به یک پردازش تخصیص داده شود. و تک تک هسته ها می توانند جهت افزایش بازدهی به کار گرفته شوند.

تعداد زیادی سیستم عامل و تجهیزات سخت افزاری برای پشتیبانی از این معماری تولید شده اند . SMP از یک سیستم عامل واحد برای به اشتراک گذاشتن منابع و حافظه بهره می گیرد. و نرم افزار ها توسط تکنیک های مالی تردینگ می توانند از این امکانات بهره گیرند.

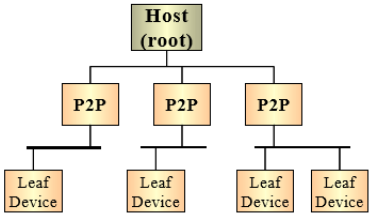
از جمله سیستم عامل هایی که با پشتیبانی از این معماری پیاده سازی شده اند، Unix و NT هستند.

AMP یا پردازش نامتوازن، بیشتر جهت انجام دادن فعالیت های مشخص و جدا توسط چند پردازنده مناسب است (مثل DSPها جهت پردازش سیگنال ها). در این معماری این مزیت وجود دارد که بر روی هر هسته یک سیستم عامل اجرا شود (مثل ترکیب nucleus و android)

PCI،PCI-X و نحوه ی پیاده سازی InfiniBand + DMA

- *Understanding PCI Bus, PCI-Express and InfiniBand Architecture, Mellanox Technologies Inc. Document Number 2006 WP*
- *Linux Device Drivers, Third Edition , Chapter 15, Section 4*

Bus از نوع PCI، در اوایل سال 1990 جهت ارائه ی یک واسط (interface) برای فراهم کردن امکان ارتقا واسط های ورودی خروجی I/O توسط کاربران یا تولید کنندگان در کامپیوتر های شخصی توسعه یافت. آخرین پیشرفت در زمینه ی توسعه ی PCI که PCI-X بود، یک واسط ۶۴ بیتی موازی با فرکانس 133MHZ و پهنای باند ۸Gb/s است. با وجود پیشرفت ها و مزیت های فراوان PCI از جمله DDR ، در صنعت کامپیوتر هزینه ی پین های 64 بیتی نسبت به پهنای باندی که فراهم می کند بسیار بالاست.



در معماری PCI ، این BUS اساسا یک واسط بین CPU و دستگاه های دیگر است و از روش P2P (PCI to PCI) جهت افزایش تعداد دستگاه های روی یک باس استفاده می شود. در بین تمام این دستگاه ها یک حافظه ی فیزیکی وجود دارد. این معماری دید یکسان به منابع سیستم دارد و با سیستم های دارای یک Master یا CPU بسیار کارامد عمل می کند.اما از آنجایی که هر دستگاه به طور مستقل Master نمی تواند یک فرآیند انتقال را انجام دهد، این معماری برای معماری های چند پردازنده ای مناسب نیست. (در معماری *InfiniBand* از روش Channel Based استفاده می شود که این محدودیت را رفع کرده و تا ۱۶ هزار دستگاه در کانال ها یا queue pair ها امکان دسترسی مستقیم به حافظه مجازی یا فیزیکی را دارند)

Feature	InfiniBand	PCI/PCI-Express
I/O Sharing	Yes	No
System Hierarchy	Channel Based Fabric	Memory Mapped Tree
Kernel Bypass	Memory Protection & Address Translation	No
System Software Interface	Transport level connections	Low Level load/store

آگوست سال قبل (از نگارش مقاله !! سال ۲۰۰۵) شرکت اینتل **PCI-Express** را به عنوان یک **ارتقا** از PCI و یک واسط چیپ به چیپ معرفی کرد. PCI-X یک واسط I/O سریال که از دو کانال ارتباطی استفاده می کند. این روش جهت برقراری ارتباط با پهنای بالا اما استفاده ی کمتر از پین های گران قیمت ۶۴ بیتی PCI بود.

تخصیص حافظه DMA

یکی از مهم ترین مشکلات این است که حافظه DMA از اندازه صفحات حافظه بیشتر است! و این آدرس باید از صفحات پشت سر هم داده شود تا توسط باس PCI منتقل شوند. در زیر یک مثال از ارتباط DMA با یک دستگاه فرضی PCI در سیستم عامل لینوکس آمده.

```
int dad_transfer(struct dad_dev *dev, int write, void *buffer, size_t count)
{
    dma_addr_t bus_addr;

    /* Map the buffer for DMA */
    dev->dma_dir = (write ? DMA_TO_DEVICE : DMA_FROM_DEVICE);
    dev->dma_size = count;
    bus_addr = dma_map_single(&dev->pci_dev->dev, buffer, count, dev->dma_dir);
    dev->dma_addr = bus_addr;

    /* Set up the device */
    writeb(dev->registers.command, DAD_CMD_DISABLEDMA);
    writeb(dev->registers.command, write ? DAD_CMD_WR : DAD_CMD_RD);
    writel(dev->registers.addr, cpu_to_le32(bus_addr));
    writel(dev->registers.len, cpu_to_le32(count));

    /* Start the operation */
    writeb(dev->registers.command, DAD_CMD_ENABLEDMA);
    return 0;
}
```

Application Program و System Program

- *IBM Knowledge Center - z/Transaction Processing Facility Enterprise Edition 1.1.0.9 - Application and system programs*

یکی از تفاوت اساسی بین برنامه های سطح سیستم و سطح اپلیکیشن :

- Application ها جهت تفسیر معنایی پیام های کاربرها استفاده می شوند. برای مثال در سیستم رزرو بلیت هواپیمایی مامور فروش بلیت که از مشتریان در خواست رزرو بلیت را دریافت می کند.
- نرم افزار های سیستمی سرویس هایی را ارائه می دهند که اپلیکیشن ها جهت استفاده از منابع سیستم به آن نیاز دارند. این سرویس های سیستمی عموما به صورت ماکرو هایی برای نرم افزار و مجزا از زبان برنامه نویسی ارائه می شوند. مثل کتابخانه ها یا API های سیستم عامل.

"A functional distinction between application and system programs is:

- Application programs are used to interpret the semantic content of messages entered by the end users of the z/TPF system. An end user is, for example, an airline reservations agent obtaining a request from a customer to reserve space on a flight.
- System programs provide system services that shield application programs from many of the details necessary for sharing system resources. The system services, in many cases, are provided in the form of macros for application program use and which, loosely, form a language. For example, to read a record from modules, the FINDC macro is an SVC I/O request to read (get) a record. "

پرسش ۲

خیر، پردازنده نمی تواند به روند اجرای برنامه ای که درخواست عملیات DMA را کرده دست بزند تا زمانی که درخواست آماده نشده و وقفه مربوط به آن از طرف DMA نیادمه باشد چون این کار روند ترتیبی و منطقی برنامه را عملا برهم می زند.

همانطور که در سوال قبلی گفته شد " کاری که دستگاه **DMA** انجام می دهد **از دید پردازنده عملا غیر قابل مشاهده است**." و فرآیند زمان بر خواندن و انتقال داده ها از دستگاه های خارجی که عموما بسیار کند است در زمان هایی که BUS حافظه جانبی توسط پردازنده مشغول نیست صورت می گیرد و این کار باعث می شود پردازنده پروسه ها و عملیات دیگر را در زمان انتقال انجام دهد تا زمانی که وقفه مربوط به DMA صدا زده شود و عملیات اجرایی برنامه ادامه پیدا کند.

پرسش ۳

- *Syscalls, Linux Man Pages (2)*
- *Syscall, Linux Man Pages (2)*

API یا **رابط برنامه‌نویسی نرم‌افزار** یک مجموعه از تابع ها، روتین ها و پروتوکل ها برای ساختن نرم افزار های کامپیوتری است که به وسیله آن از سیستم عامل یا برنامه های دیگر تقاضای سرویس می کنند.

System Call ها در واقع درخواست دسترسی برنامه ها به سرویس های سیستم عامل مثل تولید یک پروسه ی جدید ویا دسترسی به قسمت خاصی از هارد دیسک می باشند. و در اکثر سیستم عامل ها این امکان وجود دارد که در خواست های سیستمی حتی از سطح کاربری قابل اجرا باشند. البته پیاده سازی این درخواست های سیستمی به خاطر مسائل امنیتی **نیازمند تخصیص سطوح دسترسی مختلف** به نرم افزارها می باشد و سیستم عامل تمامی درخواست های داده شده از طرف هر نرم افزاری را قبول نخواهد کرد.

در مستندات سیستم عامل لینوکس، یک فراخوانی سیستمی به عنوان **"واسط بنیادی میان نرم افزار و هسته ی لینوکس"** تعریف شده است.

فراخوانی های سیستمی عموما به صورت مستقیم اجرا نمی شوند و به وسیله ی تابع های واسطه ای که در API لینوکس و کتابخانه glibc تعرف شده اند صدا زده می شوند. و اغلب و نه همیشه نام فراخوانی های سیستمی با تابعی که در این کتابخانه ها تعبیه شده یکسان است به طور مثلا truncate() تابع کرنل truncate را فراخوانی می کند. این فراخوانی ها کوتاه هستند و در طول فرآیند عملیات بسیار کوتاهی را علاوه بر کپی کردن آرگومان ها انجام می دهند. و وضعیت اجرای خود را در errno نگه داری می کنند.

مراحل انجام

Syscall() یک تابع کتابخانه ای کوچک است و فراخوانی تابعی با آن عدد را در معادل اسمبلی اجرا می کند. این تابع وضعیت رجیستر های CPU را قبل از انجام فراخوانی سیستم ذخیره می کند و بعد از اتمام و بازگشت از فراخوانی وضعیت آن ها را مجدد باز می گرداند. همچنین هرگونه خطایی که در زمان فراخوانی رخ دهد در errorno را ذخیره می کند.

به عنوان مثال فراخوانی زیر، شناسه ی منحصر به فرد پروسه ی درحال اجرا را از طریق یک فراخوانی سیستمی به دست می آورد.

```c
#include <unistd.h>
#include <sys/syscall.h>
#include <sys/types.h>

int main(int argc, char *argv[])
{
   pid_t tid;
   tid = syscall(SYS_gettid);
}
```

پرسش ۴

Batch Operating Systems

- در این سیستم عامل ها یکسری دستورات یا Job پشت سرهم و بدون نیاز به دخالت دستی اجرا می شوند. و تمامی پارامتر های زمان اجرا از قبل توسط فایل های کنترلی، آرگومان ها و .... داده می شوند.
- بسیار پیاده سازی آنها و روند اجرایی برنامه ها در آنها ساده است.
- محدودیت های بسیاری دارند و امکان تعامل با آنها وجود ندارد

Distributed Operating Systems

- در این سیستم عامل ها عملیات پردازش به صورت موازی صورت می گیرد.
- سرعت و بازدهی بسیار بالایی دارند و امکان به اشتراک گذاشتن منابع و گسترش تنوان پردازشی آسانی دارند.
- سرعت ارتباطی بین Node های مختلف در بازدهی کلی تعیین کننده است، پیچیدگی همزمانی بالایی دارند، هزینه ی بالاتری نسبت به روش های دیگر دارند.

Time-Sharing Operating Systems

- یا TSOS ها که امکان تخصیص بازه های زمانی به پروسه ها و کاربر های متفاوت را دارند.
- امکان چند کاربره بودن را دارا می باشند.
- پردازش ها نسبت به حالت قبل کند تر بوده و مشکلات به اشتراک گذاری منابع وجود دارد.

Real-Time Operating Systems

- یا RTOS ها که با دو نوع پیاده سازی Time-Sharing و Event-Driven وجود دارند. و با الگوریتم های زمان بندی امکان فراهم کردن اجرای بلادرنگ در نرم افزار ها را فراهم می کنند. از جمله Windows CE.
- از جمله بدی های اینگونه سیستم عامل ها، محدودیت تسک های هم زمان، استفاده ی بیش از اندازه از منابع سیستمی، نیاز به الگوریتم های پیچیده و برنامه نویسی دشوار می باشد.

پرسش ۵

- https://wiki.archlinux.org/index.php/Arch_Linux#History

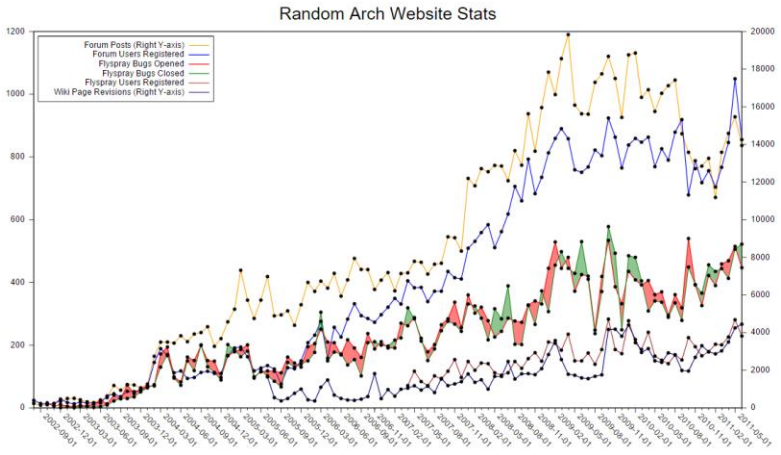
تاریخچه ی توزیع Arch Linux

سال های اولیه

Judd Vinet یک برنامه نویس کانادایی و گیتاریست (!) توسعه ی *Arch Linux* را از اوایل سال ۲۰۰۱ آغاز کرد واولین نسخه ی پخش شده 1. ArchLinux در مارچ ۲۰۰۲ عرضه شد. وینت به شدت تحت تاثیر سادگی و زیبایی توزیع های Slack ware، BSD، PLD Linux و CLUX قرار گرفته بود.ولی به خاطر عدم وجود سیستم مدیریت پکیج از آنها نا امید شده بود. وینت توزیع لینوکس خود را با الهام گرفتن از مفاهیم و اساس مشابه آنها و با توسعه ی یک سیستم مدیریت پکیج به نام *pacman* برای انجام خودکار فرآیند نصب و حذف و به روز رسانی پکیج ها ساخت.

سال های میانی

جامعه ی کاربران آرچ به طور مداوم شروع به افزایش کرد و این جامعه تحت شعار یک جامعه ی باز، دوستانه و مفید شکل گرفت. تعداد وب سایت های مرتبط تا سال ۲۰۱۱ به ۲۰۰۰۰ رسید !



طلوع Griffin !

اواخر سال ۲۰۰۷ وینت از فعالیت خود بر روی پروژه استعفا داد و کم کم این کار را به توسعه دهنده ی آمریکایی Aaron Griffin سپرد که تا به امروز این کار را به عهده دارد.در طول سالها جامعه ی آرچ به تکامل و رشد و کامل شدن خود ادامه داد و با بازنگاری ها و توجه غیرقابل باوری به عنوان یک توزیع لینوکس با توجه به حجمش قرار گرفت. برنامه نویسان آرچ هنوز بی پول مانده اند !! و به عنوان برنامه نویسان داوطلب پاره وقت و بدون هیچ برنامه ای برای پولی کردن این توزیع فعالیت می کنند.

فهرست منابع

- *DMA Fundamentals on Various PC Platforms, from A. F. Harvey and Data Acquisition Division Staff NATIONAL INSTRUMENTS*
- *http://wiki.osdev.org/ISA_DMA*
- *ece.ubc.ca, ELEC 379: DESIGN OF DIGITAL AND MICROCOMPUTER SYSTEMS 1998/99 WINTER*
- *Lina J. Karam, Ismail AlKamal, Alan Gatherer, Gene A. Frantz, David V. Anderson, Brian L. Evans (2009). "Trends in Multi-core DSP Platforms" (PDF). IEEE Signal Processing Magazine, Special Issue on Signal Processing on Platforms with Multiple Cores.*
- *Mentor Graphics Blog, AMP vs SMP*
- *Understanding PCI Bus, PCI-Express and InfiniBand Architecture, Mellanox Technologies Inc. Document Number 2006 WP*
- *Linux Device Drivers, Third Edition , Chapter 15, Section 4*
- *IBM Knowledge Center - z/Transaction Processing Facility Enterprise Edition 1.1.0.9 - Application and system programs*
- *Syscalls, Linux Man Pages (2)*
- *Syscall, Linux Man Pages (2)*
- https://wiki.archlinux.org/index.php/Arch_Linux#History