

Operační systémy 1









Úvod do operačních systémů

Petr Krajča



Katedra informatiky
Univerzita Palackého v Olomouci

- email: petr.krajca@upol.cz
- přednáška: čtvrtrek 9:45 – 11:15
- cvičení
 - programování na úrovni procesoru (assembler)
 - offline, podklady na webu předmětu
 - odevzdávání vybraných úkolů (email); vztahuje se i na neúplné řešení
 - konzultace
- zápočet
 - pokud dojde k obnovení prezenční výuky: písemná práce na konci semestru
 - jinak: naprogramování zadaných úkolů
- konzultační hodiny: emailem nebo po domluvě
- www: <http://phoenix.inf.upol.cz/~krajcap/courses/2021LS/OS1/>
- podmínky pro udělení zápočtu budou sděleny na cvičení

-  Keprt A. *Operační systémy. 2008*
-  Keprt A. *Assembler. 2008*
-  Silberschatz A., Galvin P.B., Gagne G. *Operating System Concepts, 7th Edition*. John Wiley & sons, 2005. ISBN 0-471-69466-5.
-  Tanenbaum A.S. *Modern Operating Systems, 2nd ed*. Prentice-Hall, 2001. ISBN 0-13-031358-0.
-  Stallings, W. *Operating System Internals and Design Principles, Fifth Edition*. Prentice Hall, 2004. ISBN 0-13-127837-1.
-  Bovet D., Cesati M. *Understanding Linux Kernel, 3rd ed*. O'Reilly, 2006. ISBN 0596005652.
-  Solomon D.A., Russinovich M. E. *Windows Internals: Covering Windows Server 2008 and Windows Vista*. Microsoft Press, 2009. ISBN 0735625301.
-  Jelínek L. *Jádro systému Linux: kompletní průvodce programátora*. Brno, Computer Press, 2008.

- **abstrakce HW**
- vyvíjet **software na míru jednoho HW náročné/neefektivní** (obvykle); hardware je neuvěřitelně složitý
- operační systém – **rozhraní mezi HW a SW**
- **operační systém poskytuje abstrakci nad daným hardwarem** (+ jazyky vyšší úrovně)
- v konečném důsledku **několik úrovní abstrakce**

Vrstvy HW/SW

- 1 hardware
 - 2 operační systém (OS)
 - 3 standardní knihovna (libc, CRT)
 - 4 systémové nástroje
 - 5 aplikace
- jádro OS vs. aplikace
 - hranice mezi vrstvami nemusí být ostré
 - situace se komplikuje – virtualizace, běhová prostředí
 - další funkce: **operační systém zajišťuje správu zdrojů** – sdílení času (CPU, zařízení), místa (paměť, disky)

- 1. generace (1945 – 1955): relé, elektronky a program „zadrátovan“ do počítače
- 2. generace (1955 – 1965): tranzistory, děrné štítky, dávkové zpracování a FORTRAN
- 3. generace (1965 – 1980): integrované obvody, IBM System/360 a minipočítače PDP
 - multitasking
 - timesharing (CTSS – MIT)
 - současná práce více uživatelů, ale pořád prvky dávkového zpracování
 - spooling (sdílení periférií)
 - virtuální paměť; první síť
- 4. generace (1980 – současnost): vysoký stupeň integrace; Intel 8080, x86; CP/M, DOS, Windows 95/NT, Unix, GNU/Linux
- 5. generace (1990 – současnost): mobilní OS; Android, iOS

Podle určení

- Mainframy – OS/400, zOS
- Serverové/Multiprocesorové – *BSD, AIX, GNU/Linux, HP-UX, Solaris, Windows NT, ...
- Desktopové – *BSD, GNU/Linux, macOS, Windows NT, ...
- Realtime – VxWorks, QNX, ...
- Distribuované
- Mobilní zařízení – Android, Bada, Blackberry OS, iOS, Symbian, Windows Phone, WebOS, Windows 10 Mobile, ...
- Experimentální/výukové – Minix, Plan 9

Historické záležitosti

- CP/M, MS-DOS, Windows 9x
- BeOS, Mac OS (classic), OS/2

John von Neumannova architektura

- CPU (ALU, řadič)
- paměť **společná pro program i data** (vs. harvardská architektura)
- vstup/výstup
- sběrnice (řídící, adresní, datová)
- instrukce procesoru jsou zpracovávány v řadě za sebou (není-li uvedeno jinak)

Obecná struktura CPU

- Aritmeticko-logická jednotka (ALU) – provádí výpočty
- řídící jednotka – řídí chod CPU
- registry – slouží k uchování právě zpracovávaných dat (násobně rychlejší přístup než do paměti); speciální registry obsluhující chod CPU: IP (instruction pointer), program status word (PSW, FLAGS), IR (instruction register), SP (stack pointer)

Instrukční sada (ISA)

- sada instrukcí ovládající procesor (specifická pro daný CPU/rodinu CPU)
- instrukce a jejich operandy jsou reprezentovány jako čísla \Rightarrow strojový kód
- každá instrukce má obvykle 0 až 3 operandy (může to být registr, konstanta nebo adresa místa v paměti)
- pro snazší porozumění se instrukce CPU zapisují v jazyce symbolických adres (assembleru)

00000000 <main>:

0:	8b 4c 24 04	mov	ecx,DWORD PTR [esp+0x4]
4:	b8 01 00 00 00	mov	eax,0x1
9:	83 f9 00	cmp	ecx,0x0
c:	0f 8e 0a 00 00 00	jle	1c <main+0x1c>
12:	f7 e9	imul	ecx
14:	83 e9 01	sub	ecx,0x1
17:	e9 ed ff ff ff	jmp	9 <main+0x9>
1c:	c3	ret	

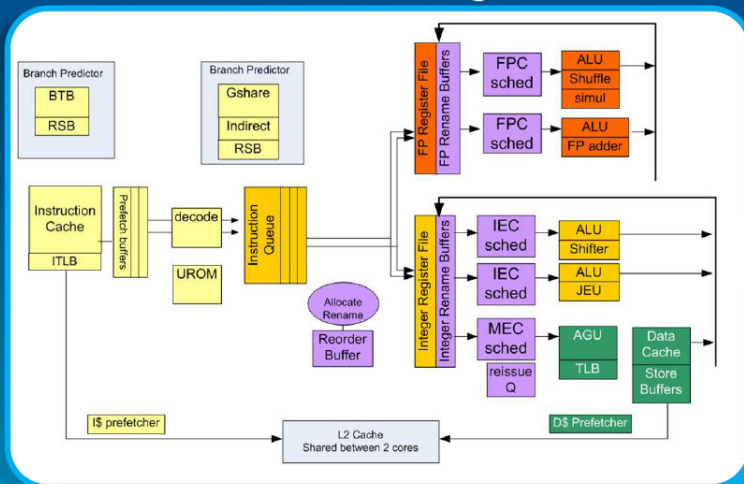
00000000 <main>:

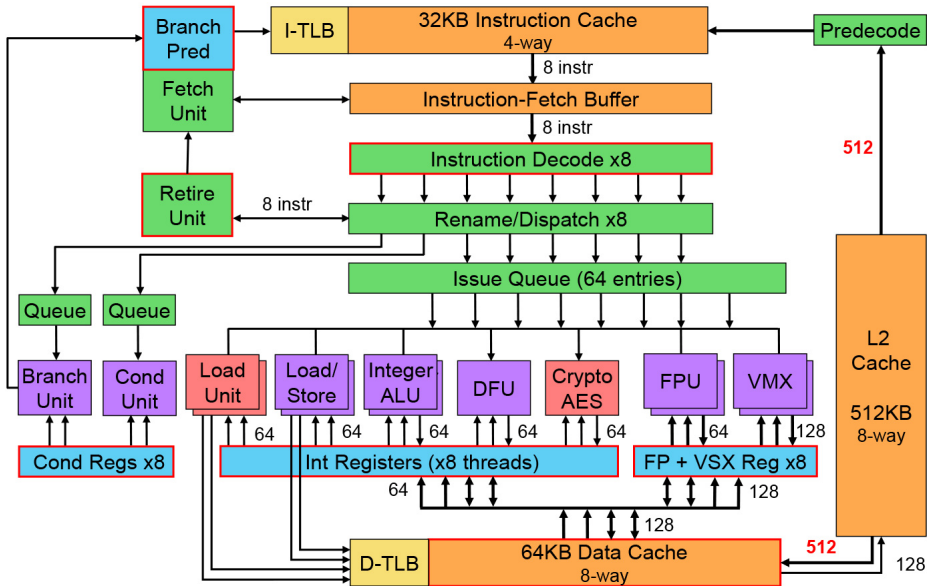
0:	9d e3 bf 88	save %sp, -120, %sp
4:	a0 10 20 01	mov 1, %l0
8:	80 a6 00 00	cmp %i0, %g0
c:	04 80 00 06	ble 24 <main+0x24>
10:	01 00 00 00	nop
14:	a0 5c 00 18	smul %l0, %i0, %l0
18:	b0 26 20 01	dec %i0
1c:	10 bf ff fb	b 8 <main+0x8>
20:	01 00 00 00	nop
24:	b0 10 00 10	mov %l0, %i0
28:	81 c7 e0 08	ret
2c:	81 e8 20 00	restore

- instrukce jsou zpracovávány v několika krocích:
 - 1 načtení instrukce do CPU (Fetch)
 - 2 dekodování instrukce (Decode)
 - 3 výpočet adres operandů
 - 4 přesun operandů do CPU
 - 5 provedení operace (Execute)
 - 6 uložení výsledku (Write-back)
- **pipelining** – umožňuje zvýšit efektivitu CPU
- superskalární procesory – procesor může mít víc jednotek např. pro výpočty (FPU, ALU)
- je potřeba zajistit správné pořadí operací
- **synchronizace**, **problém s podmíněnými skoky** (branch prediction)
- Simultaneous multithreading (SMT): zpracování instrukcí více vláken v jednom cyklu

Instr. No.	Pipeline Stage						
	IF	ID	EX	MEM	WB		
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Core Block Diagram





- registry jsou 32bitové
- obecně použitelné (i když existují určité konvence, jak by se měli používat)
 - EAX (Accumulator) – střadač pro násobení a dělení, vstupně-výstupní operace
 - EBX (Base) – nepřímá adresace paměti
 - ECX (Counter) – počítadlo při cyklech, posuvech a rotacích
 - EDX (Data)
- každý registr má svou spodní 16bitovou část reprezentovanou jako registr AX, BX, CX, DX
- tyto 16 bitové registry lze rozdělit na dvě 8bitové části reprezentované jako AH, AL, BH, BL, ...

Další registry

- **EDI (Destination Index)** – adresa cíle
- **ESI (Source Index)** – adresa zdroje
- **EBP (Base Pointer)** – adresace parametrů funkcí a lokálních proměnných
- **ESP (Stack Pointer)** – ukazatel na vrchol zásobníku (adresa vrcholu zásobníku)
- **EIP (Instruction Pointer)** – ukazatel na aktuální místo programu, adresa instrukce následující za právě prováděnou instrukcí, není možné jej přímo měnit (jen patřičnými instrukcemi)
- **EF(LAGS)** – příznaky nastavené právě proběhlou instrukcí
- spodních 16 bitů těchto registrů lze adresovat pomocí registrů DI, SI, BP, SP, IP, F(LAGS); další dělení není možné
- **ESI a EDI jde používat jako obecně použitelné**
- změny v registrech **EBP, ESP by měly být uvážené**

- operandy instrukcí mohou být

- r – registry
- m – adresa místa v paměti
- i – přímé hodnoty (konstanty)

- paměť lze v jedné instrukci adresovat pouze jednou

MOV r/m, r/m/i	; $op1 := op2$
ADD r/m, r/m/i	; $op1 := op1 + op2$
SUB r/m, r/m/i	; $op1 := op1 - op2$
NEG r/m	; $op1 := -op1$
MUL r/m	; $EDX:EAX := EAX * op1$
IMUL r, r/m	; $op1 := op1 * op2$
IMUL r, r/m, i	; $op1 := op2 * op3$
OR r/m, r/m/i	; $op1 := op1 \vee op2$
AND r/m, r/m/i	; $op1 := op1 \wedge op2$
XOR r/m, r/m/i	; $op1 := op1 \hat{~} op2$
NOT r/m	; $op1 := \sim op1$

; do registru EAX uloži obsah EBX

```
mov eax, ebx
```

; převrati spodních 16 bitů v registru ECX

```
xor ecx, 0x0000ffff
```

; přičte k registru cx hodnotu registru si

```
add cx, si
```

; takto nejde -- nesedí velikosti registru

```
add ecx, si
```

; vyneguje obsah registru edx

```
neg edx
```

INC r/m ; $op1 := op1 + 1$

DEC r/m ; $op1 := op1 - 1$

SHL r/m, i ; $op1 := op1 \ll op2$ (neznaménková operace)

SAL r/m, i ; $op1 := op1 \ll op2$ (znaménková operace)

SHR r/m, i ; $op1 := op1 \gg op2$ (neznaménková operace)

SAR r/m, i ; $op1 := op1 \gg op2$ (znaménková operace)

ROL r/m, i ; *rotace bitů doleva*

ROR r/m, i ; *rotace bitů doprava*

- místo přímé hodnoty (konstanty) lze použít registr CL

- jednotlivé operace nastavují hodnoty bitů v registru EF(lags)
- ne všechny instrukce mění všechny příznaky
- registr EF mj. obsahuje příznaky:
 - SF (sign flag) – nastaven, pokud je výsledek záporný
 - ZF (zero flag) – výsledek byl nula
 - CF (carry flag) – nastaven, pokud při operaci došlo k přenosu mezi řády
 - OF (overflow flag) – příznak přetečení mimo daný rozsah hodnot
- některé další příznaky:
 - TF (trap flag) – slouží ke krokování
 - DF (direction flag) – ovlivňuje chování instrukcí blokového přesunu
 - IOPL (I/O privilege level) – úroveň oprávnění (2 bity, pouze jádro)
 - IF (Interrupt enable flag) – možnost zablokovat některá přerušení (pouze jádro)