Interaktyvioji multiprograminė
128bitų NTFS operacinė sistema (emu128)

(papildomai gali būti palaikomos 64 ir 256 bitų komandos/operandai/registrai

bei ZFS failų sistema(darbui su > 2TiB adresavimo aplinka))

I DALIS – REALI IR VIRTUALI MAŠINA- APRAŠYMAS IR GALIMYBĖS

II DALIS – MULTIPROGRAMINĖ OPERACINĖ SISTEMA

Dokumentacijos versija v1.0.500423

Parengė Kęstutis Matuliauskas

Turinys

Turinys

Įvadas

Aprašymas:

emu128 – tai 128 bitų sistemos emuliatorius, veikiantis virtualios sistemos pagrindu naudodamasis realios OS resursais. emu128 įtraukia tik bazines Core i7, AMD Buldozer, Intel Sandy Bridge, Intel Itanium.

Pagrindinės savybės:

- **Baziniai registrai**(AX,BX,CX,DX,SP,..): 8,16,32,64,128 bitų.
- **Multimedija registrai**(MMX): 64 bitų.
- **Išplėstiniai SSE registrai**(XMM): 128 bitų.
- **Vektorinės registrai**(YMM): 256 bitų.
- **Failų sistemos:**
  - NTFS *[klasterio dydis 4kB]* : 2GB -> 2TB
  - ZFS: iki 256UiB
- **Procesorius:** 128 bitų, galintis adresuoti iki 2^{128}B atminties
- **Adresuojama atmintis:** iki 2^{128}B (256UiB)
- **Transliatorius VM->RM:** Microsoft Visual C# Compiler
- **Realūs CPU:** x86-128 arba IA-128
- **Taip pat palaikomi realūs CPU:** x86-64*(dalinai)*, IA-64 *(dalinai)*

Realios OS savybės pagal nutylėjimą:

- Atmintį sudaro 2^{36} Double-Quadruple žodžių*(64Gi*16B=1 TiB)*.
- Kiekvieną žodį sudaro 128 bitai *(8*16b – x86-128 ir 128b IA-128).*

Virtualios OS savybės pagal nutylėjimą:

- Atmintį sudaro 2^{32} Double-Quadruple žodžių*(64Mi*16B=1 GiB)*.
- Kiekvieną žodį sudaro 128 bitai *(8*16b – x86-128 ir 128b IA-128).*
- Realizuoti VISUS RM procesoriaus registrus, išskyrus MSR(*Model-specific registers*).
- Realizuoti visas pagrindines x86 instrukcijas, būtinas norint parodyti kiek galima maksimalias x86-128 galimybes.

Segmentuotame IA-32 režime(limitus galima pakeisti):

- Steko segmentui yra skiriami pirmi 256 žodžiai.
- Duomenų segmentui yra skiriami pirmi

Puslapiuotame IA-32e režime(limitus galima pakeisti):

Segmentacija neegzistuoja, vietoje jos, naudojamas puslapiavimas:

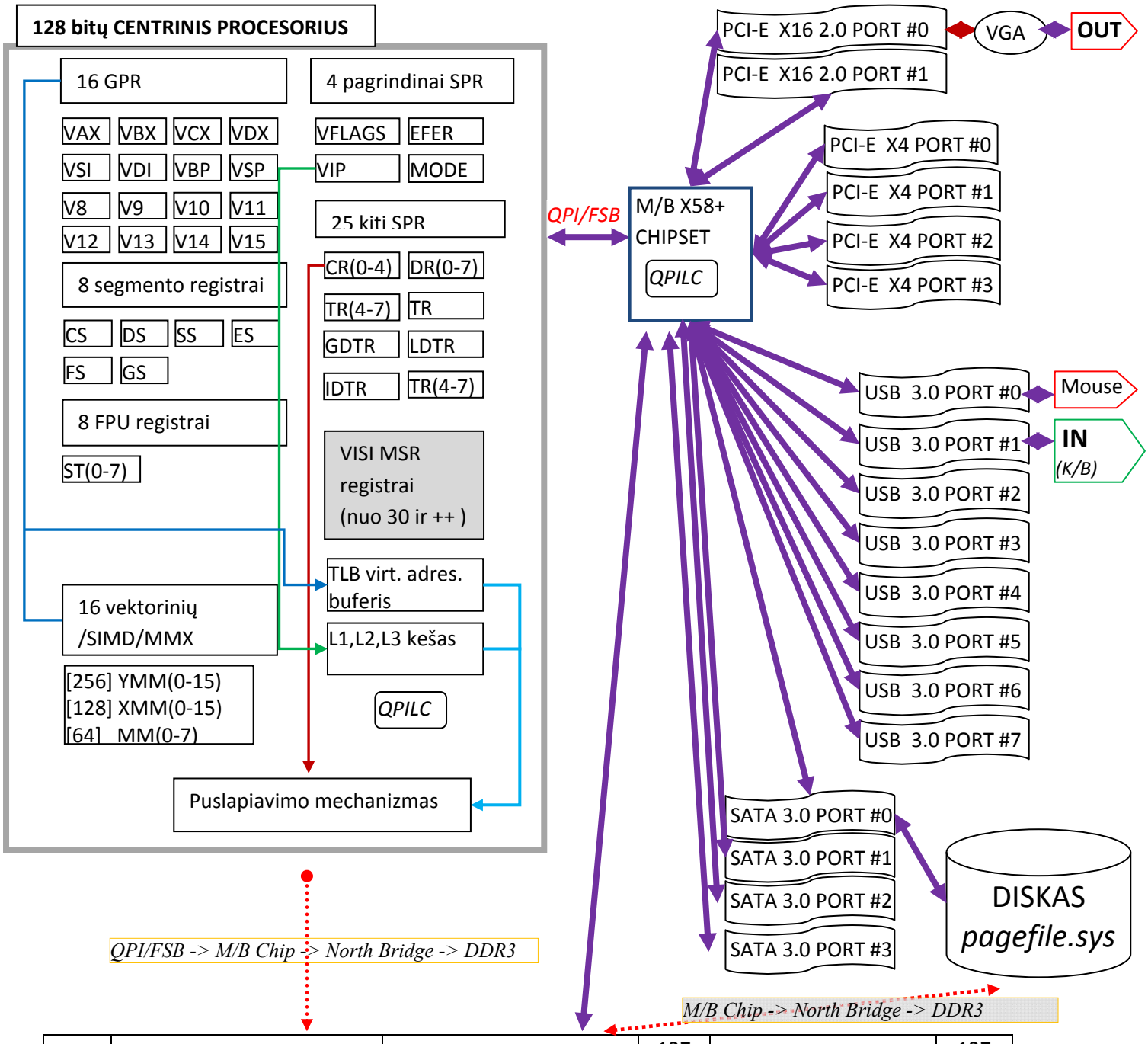
- Stekui skiriami pirmi 8 puslapiai po 4KiB.
- Duomenys ir kodas dalinasi likusią paskirtą dalį puslapiuose po 1GiB arba 2MiB.

Realios ir virtualios mašinos modeliai

Trumpiniai:

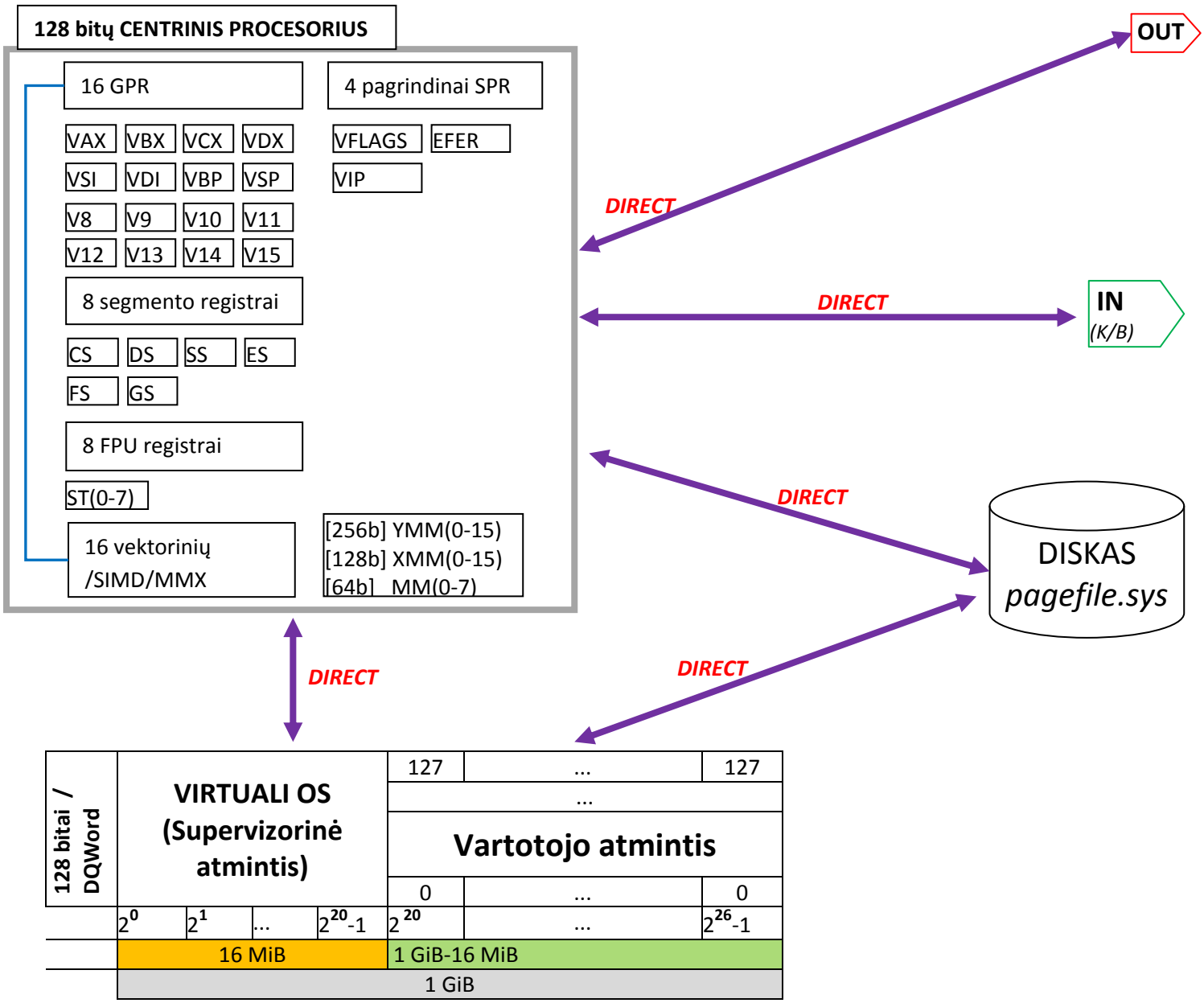
QPI – **Q**uick**P**ath **I**nterconnect
QPILC – **QPI** **L**ink **C**ontroller

Realios mašinos modelis:



128 bitai / DQWord	BIOS/MBR	REALI OS (Supervizorinė atmintis)	127 ... 127 ... Vartotojo atmintis 0 ... 0
	2^0 2^1 ... $2^{16}-1$	2^{16} 2^1 ... 2^N-1	2^N ... $2^{36}-1$
	1 MiB	~384MiB→4 GiB+	>= 1,5 GiB
	1 TiB		

Virtualios mašinos modelis:



Realios ir Virtualios mašinų registrai

Trumpiniai:

GPR - Global Purpose Registers *(bendros paskirties registrai)*
SPR – Miscellaneous/**S**pecial **P**urpose **R**egisters *(specialios paskirties registrai)*
IH – **I**nterrupt **H**andler *(pertraukimų mechanizmo valdiklis)*
TPR – **T**est **P**urpose **R**egisters *(tikrinimo paskirties registrai)*
PFLA - **P**age **F**ault **L**inear **A**ddress *(puslapiavimo klaidos fizinis adresas)*
PDPR – **P**age **D**irectory **B**ase **R**egister *(puslapiavimo katalogo bazinis registras)*
MCE - **M**achine **C**heck **E**xceptions
FPU – Floating **P**oint **U**nit *(slankaus kablelio įrenginys, x86 architektūroje - integruotas į procesorių, modernūs procesoriai Jų turi 128 vienetus ir daugiau)*
ALU – Artmetic **L**ogic **U**nit *(aritmetinis loginis įrenginys, integruotas į procesorių, modernūs procesoriai Jų turi 128 vienetus ir daugiau)*

Bendros paskirties registrai (*GPR)*:

(tačiau juos nebūtinam naudoti pagal rekom. paskirtį)

	Registrų grupė ir pogrupis	Rekomend. paskirtis	Registro originalus pavadinimas	Registro kodas	Vyr. baitas *(High Byte) ?H*	Jaun. baitas *(Low Byte) ?L*
BENDROS PASKIRTIES REGISTRAI	**Bendros paskirties registrai:**					
	Akumuliatorius		**A**ccumulator	**AX**	A**H**	A**L**
	Bazės indeksas *(pvz. sekose)*		**B**ase index *("arrays")*	**BX**	B**H**	B**L**
	Skaitliukas		**Counter**	**CX**	C**H**	C**L**
	Duomenys		**Data/General**	**DX**	D**H**	D**L**
	Indeksų registrai:					
	Šaltinio indeksas *(darbui su eilutėmis)*		**S**ource **I**ndex *(for "strings" operations)*	**SI**	*nėra*	SI**L**
	Tikslo indeksas *(darbui su eilutėmis)*		**D**estination **I**ndex *(for"strings" operations)*	**DI**	*nėra*	DI**L**
	Rodyklės:					
	Steko rodyklė (ne steko operacijose) *(ne pertraukimo režime)*		**S**tack **P**ointer *(non-interrupt level)*	**SP**	---	SP**L**
	Steko bazės rodyklė		Stack **B**ase **P**ointer *(Frame Pointer)*	**BP**	---	VB**L**
	Papildomi bendros paskirties registrai:					
	Papildomi išplėstiniai registrai *(128 bitų SSE ir 256 bitų vektoriniai)*		**R**egister **E**xtension No.8-15 **V**ector **E**xtension No.8-15	**R***8-15*, **V***8-15*	---	R*8-15***B**, V*8-15***B**
	Segmentų registrai: (TIK IA-32/x86-16/x86-32)					
	Kodo segmentas		**C**ode **S**egment *(Text Segment)*	**CS**	---	---
	Duomenų segmentas *(4 lygis, žemiausias)*		**D**ata **S**egment	**DS**	---	---
	Steko segmentas		**S**tack **S**egment	**SS**	---	---
	Papildomas duomenų segmentas *(3 lygis)*		**E**xtra Data **S**egment *(Extra Register, Destination of string operation)*	**ES**	---	---
	Papildomas duomenų segmentas *(2 lygis)*		**F**lat Data **S**egment *(Extra Register, (thread local storage in Windows)*	**FS**	---	---
	Papildomas duomenų segmentas *(1 lygis, aukščiausias)*		**G**lobal Data **S**egment *(Extra Register, generally unused)*	**GS**	---	---

Specialios paskirties registrai (*SPR)*:

(registrai, turintys konkrečią paskirtį ir naudojami tik pagal ją)

Registrų grupė ir pogrupis	Funkcinė registro paskirtis	Registro originalus pavadinimas	Registro kodas
SPECIALIOS PASKIRTIES REGISTRAI			
Rodyklių registrai:			
	Einamos komandos rodyklė (Saugo programos skaitliuką, einamos instrukcijos adresą)	**I**nstruction **P**ointer *(Program counter)*	IP
	Steko rodyklė (operacijose su steku) *(gražina esamos programos adresą įrašytą į steko pertraukimo lygmenyje)*	**S**tack **P**ointer *(interrupt hander level)*	SP
	Būsenų registras	Program **FLAGS** *(Status Register)*	**FLAGS**
Valdymo registrai:			
	Pagrindinių procesoriaus operacijų valdymo būsenų*(FLAGS)* registras	**C**ontrol **R**egister No.**0**	**CR0**
	Rezervuotas	**C**ontrol **R**egister No.**1** *(reserved)*	**CR1**
	Įvykus klaidai puslapiavimo mechanizme, išsaugo adresą, kurį programą bandė pasiekti	**C**ontrol **R**egister No.**2** *(for "PFLA" value)*	**CR2**
	(kai virtualaus adresavo režimas įjungtas) Išverčia virtualius adresus į fizinius adresus *(nurodo puslapiavimo katalogą ir pusl. lenteles esamai užduočiai)*	**C**ontrol **R**egister No.**3** *(virtual addr =1, PG dir & PG tables for current task, upper 20bits - PDPR)*	**CR3**
	(kai „saugus režimas" įjungtas) Įv./išvesties stabdymo taškų valdymas, ‚hardware' pertraukimų valdymas ir kt.	**C**ontrol **R**egister No.**4** *(protected mode=1, MCE, I/O breakpoints, page size ext.)*	**CR4**
		Control **R**egister No.**8** *(64-bit mode only)*	**CR8**
		e**X**tended **C**ontrol **R**egister No.**8** *(XFEATURE_ENABLED_MASK)*	**XCR0**
		Task Priority Register	**TPR**
	MSR valdymo registrai(tik x86-64+/IA-64+)		
	Papildomo registrų valdymo registras *(leidžia įjungti SYSCALL/SYSRET ir kitus „model-specific" registrus)*	**E**xtended **F**eature **E**nable **R**egister *(for MSR)*	**EFER**
Derinimo registrai:			
	Derinimo registrai *(klaidų ieškojimo režime)*	**D**ebug **R**egister	**DR***0-3*
	Atsarginiai derinimo registrai *(Rezervuoti ateičiai)*	**D**ebug **R**egister *(reserved)*	**DR***4-5*
	Derinimo būsenos registras	**D**ebug Status	**DR6**
	Derinimo valdymo registras	**D**ebug Control	**DR7**
Tikrinimo registrai:			
	Pagrindiniai tikrinimo registrai	**T**est **R**egister	**TR***4-5*
	Tikrinimo paskirties registrai: *(TPR)*		
	Tikrinamos komandos registras	**T**est Command **R**egister	**TR***6*
	Tikrinamų duomenų registras	**T**est Data **R**egister	**TR***7*
Atminties-valdymo registrai: *(Memory-management registers)*			
	Saugo globalaus deskriptoriaus lentelės fizinį adresą	**G**lobal **D**escriptor **T**able **R**egister	**GDTR**
	Saugo vietinio deskriptoriaus lentelės fizinį adresą	**L**ocal **D**escriptor **T**able **R**egister	**LDTR**
	Saugo pertraukimų deskriptoriaus lentelės fizinį adresą	**I**nterrupt **D**escriptor **T**able **R**egister	**IDTR**
	Užduočių registras	**T**ask **R**egister	**TR**

Bendros paskirties registrai x86-128 kompiuterių architektūroje:

Rekomend. paskirtis	**0-127 bitai** *V?X, ?I, ?P, V?*	**0-63 bitai** *R?X, ?I, ?P, R?*	**0-31 bitai** *E?X, ?I, ?P, R?D*	**0-15 bitai** *?X, ?I, ?P, R?W*	**8-15 bitai** *?H*	**0-7 bitai** *?L, R?B*
Akumuliatorius	**VAX**	RAX	**E**AX	AX	A**H**	A**L**
Bazės indeksas	**VBX**	RBX	**E**BX	BX	B**H**	B**L**
Skaitliukas	**VCX**	RCX	**E**CX	CX	C**H**	C**L**
Duomenys	**VDX**	RDX	**E**DX	DX	D**H**	D**L**
Šaltinio indeksas	**VSI**	RSI	**E**SI	SI	---	SI**L**
Tikslo indeksas	**VDI**	RDI	**E**DI	DI	---	DI**L**
Steko rodyklė	**VSP**	RSP	**E**SP	SP	---	SP**L**
Steko bazės rodyklė	**VBP**	RBP	**E**BP	BP	---	BP**L**
Papildomas	**V8**	R8	R8**D**	R8**W**	---	R8**B**
Papildomas	**V9**	R9	R9**D**	R9**W**	---	R9**B**
Papildomas	**V10**	R10	R10**D**	R10**W**	---	R10**B**
Papildomas	**V11**	R11	R11**D**	R11**W**	---	R11**B**
Papildomas	**V12**	R12	R12**D**	R12**W**	---	R12**B**
Papildomas	**V13**	R13	R13**D**	R13**W**	---	R13**B**
Papildomas	**V14**	R14	R14**D**	R14**W**	---	R14**B**
Papildomas	**V15**	R15	R15**D**	R15**W**	---	R15**B**

Segmentų registrai (tik IA-32, x86-16 ir x86-32, naujesnėse – tik *„Compatibility"* režime):

Rekomend. paskirtis	**0-15 bitai**
Kodo segmentas	**CS**
(L4) Duomenų segmentas	**DS**
Steko segmentas	**SS**
(L3) Papildomas duomenų segmentas	**ES**
(L2) Papildomas duomenų segmentas	**FS**
(L1) Papildomas duomenų segmentas	**GS**

Specialios paskirties pagrindiniai registrai:

Rekomend. paskirtis	**0-127 bitai**	**0-63 bitai**	**0-31 bitai**	**0-15 bitai**
Einamos komandos rodyklė	**VIP**	**R**IP	**E**IP	IP
Būsenų registras	**VFLAGS**	**R**FLAGS	**E**FLAGS	FLAGS

Slankaus kablelio čipo*(FPU)* registrai:

Rekomend. paskirtis	**0-79 bitai**
FPU Steko viršūnė	**ST0**
FPU Steko 1-asis elementas	**ST1**
FPU Steko 2-asis elementas	**ST2**
FPU Steko 3-asis elementas	**ST3**
FPU Steko 4-asis elementas	**ST4**
FPU Steko 5-asis elementas	**ST5**
FPU Steko 6-asis elementas	**ST6**
FPU Steko galas	**ST7**

Trumpiniai:

AVX – **A**dvanced **V**ector e**X**tensions *(256 bit)*
VEX = **V**ector **EX**tensions *(256 bit)*
REX - **R**egister **EX**tensions *(64 bit)*
SSE - **S**treaming **S**IMD **E**xtensions *(128 bit)*
MMX – **M**ulti**M**edia e**X**tensions *(128 bit)*
SIMD – **S**ingle **I**nstruction, **M**ultiple **D**ata *(x86-128)*
MISD – **M**ultiple **I**nstruction, **S**ingle **D**ata *(IA-128)*
AES – **A**dvanced **E**ncryption **S**tandard
CLMUL – **C**arry-**L**ess **MUL**tiplication
FMA – **F**used **M**ultiply-and-**A**dd
AMD – **A**dvanced **M**icro **D**evices
Intel - **INT**egrated **EL**ectronics Corporation

Išplėstiniai 64, 128 ir 256 bitų registrai*(AVX,SSE,MMX)*:

0-255 bitai *(AVX , SSE5)*	**0-127 bitai** *(SSE2, 128/64-bit mode)*	**0-127 bitai** *(SSE,32-bit mode)*	**0-63 bitai** *(MMX)*
YMM0	**XMM0**	**XMM0**	MM0
YMM1	**XMM1**	**XMM1**	MM1
YMM2	XMM2	XMM2	MM2
YMM3	XMM3	XMM3	MM3
YMM4	XMM4	XMM4	MM4
YMM5	XMM5	XMM5	MM5
YMM6	XMM6	XMM6	MM6
YMM7	XMM7	XMM7	MM7
YMM8	XMM8	---	---
YMM9	XMM9	---	---
YMM10	XMM10	---	---
YMM11	XMM11	---	---
YMM12	XMM12	---	---
YMM13	XMM13	---	---
YMM14	XMM14	---	---
YMM15	XMM15	---	---

Papildomi instrukcijų rinkiniai x86/IA-64+ procesoriuose:

I.R.	Streaming SIMD Extensions																		
	ALU	FPU	MMX	3DNow!	3DNow! +	SSE	SSE2	SSE3	SSSE3	SSE4			SSE5			AES	CLMUL	FMA3	AVX
Proc.										4.1	4.2	4a	XOP	FMA4	CVT16				
Intel	1982	1982	1993	---	---	2000	2000	2004	2008	2008	2008	---		---		2008	2008	2011	2010
AMD	1982	1982	1993	1997	1999				2011	---	---	2007	2011	2011	2011	2011	2011	---	2011

Pagrindiniai registrai

	VFLAGS																														
		RFLAGS																													
			EFLAGS																												
																		FLAGS													
Bitas	127-64	63-32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13-12	11	10	9	8	7	6	5	4	3	2	1	0
Flag	-	-	-	-	-	-	-	-	-	-	-	-	ID	VIP	VIF	AC	VM	RF	0	NT	IOPL	OF	DF	IF	TF	SF	ZF	0	AF	0	PF	1	CF

Intel x86 FLAGS Registras

Bit #	Santr.	Aprašymas	Grupė
FLAGS			
0	CF	Carry flag	S
1	1	Rezervuota	
2	PF	Parity flag	S
3	0	Rezervuota	
4	AF	Adjust flag	S
5	0	Rezervuota	
6	ZF	Zero flag	S
7	SF	Sign flag	S
8	TF	Trap flag (single step)	X
9	IF	Interrupt enable flag	X
10	DF	Direction flag	C
11	OF	Overflow flag	S
12, 13	IOPL	I/O privilege level (286+ only)	X
14	NT	Nested task flag (286+ only)	X
15	0	Rezervuota	
EFLAGS			
16	RF	Resume flag (386+ only)	X
17	VM	Virtual 8086 mode flag (386+ only)	X
18	AC	Alignment check (486SX+ only)	X
19	VIF	Virtual interrupt flag (Pentium+)	X
20	VIP	Virtual interrupt pending (Pentium+)	X
21	ID	Identification (Pentium+)	X
22-31	0	Rezervuota	
RFLAGS			
32-63	0	Rezervuota	
VFLAGS			
64-127	0	Rezervuota	

S: Status flag; C: Control flag; X: System flag

RM palaikomos kompiuterių architektūros

Trumpiniai:

bit – **BI**NARY DIGI**T** *(dvejeitainis skaičius)*
Byte –**B**INAR**Y** TABL**E** *(dvejetainė lentelė - aštuoni dvejetainiai skaičiai)*
nibble – pusbaitis (*4b*)
b – bitas *(bit)*
B – Baitas *(Byte, 1B=8b)*
w – žodis (word, *single)*
dw – dvigubas žodis *(***dword,** *double)*
qw – keturgubas žodis **(qword,** *quadruple)*
ow/dqw – aštungubas žodis **(oword / dqword (*Intel:** *octuple / **Microsoft:** double quadruple)*
d – dešimtainiai skaitmenys *(decimal, 1d=4b)*

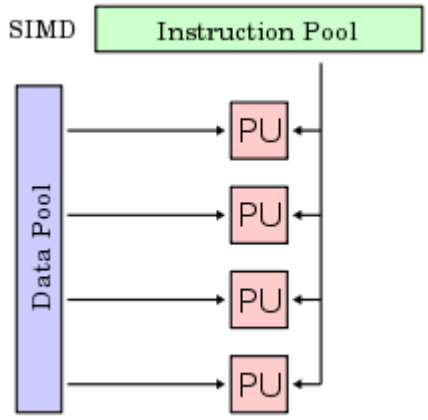
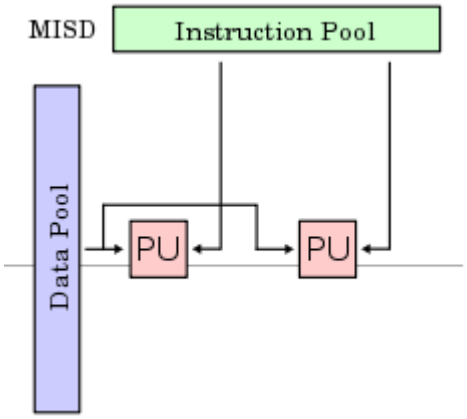
Žodžio ilgiai:

(priklauso nuo konkrečios architektūros)

x86*(i386,i486)* architektūroje *(Intel 32, AMD32)*: 2* *16 bitų / 2 Baitai*
Intel Pentium 4(SSE2), AMD Athlon XP
x86-64 architektūroje *(Intel 64, AMD64)*: 4* *16 bitų / 2 Baitai*
Susijusę procesoriai: Pentium 4 Prescott(SSE3) Intel Core 2 Penryn(SSE4), AMD Athlon 64

x86-128 architektūroje *(Intel Larabee(2011-2012), AMD Oriochi(2011))*: 8* *16 bitų / 2 Baitai*
Susijusę procesoriai:Paremti Intel Sandy Bridge(su AVX) technologija, AMD Buldozzer(SSE5)

Intel Itanium*(tar. „ay'teyniəm")* architektūrose:
IA-64: 64 bitai *(8 B)*
IA-128: 128 bitai *(16 B)*
Susiję procesoriai: Šiuo metu nepaskelbta

„x86-128" 128 bitų SSE (SIMD) realizacija	**„IA-128" 128 bitų SME (MISD) realizacija**
	
***SIMD** - Single Instruction, Multiple Data*	***MISD** - Multiple Instruction, Single Data*
Tipas, pliusai, minusai: • Operuojama kaip 16 bitų architektūra su multiplikatoriumi *(x1,x2,x4,x8)*. • - Vienai instrukcijai nuskaityti gali reikėti iki 17-18 operacijų(taktų) • + Mažesni atminties poreikiai • + Vienodai greitai supranta visas – 16, 32, 64 ir 128 bitų instrukcijas, operandus ir komandas	**Tipas, pliusai, minusai:** • Operuojama kaip 128 bitų architektūra be multiplikatoriaus. • +Vienai instrukcijai nuskaityti užtenka 1 operacijos(ciklo) • + Didesni sistemos atminties ir resursų poreikiai • + Neįmanomas, arba labai lėtas žemesnių nei 128 bitų instrukcijų ir operandų ir kitų komandų

Metai	Kompiuterių architektūra	Žodžio ilgis *w*	Sveikų skaičių ilgis *(Integer sizes)*	Slankaus kablelio skaičių ilgis *(Floating Point sizes)*	Instrukcijų ilgis *(Instruction Sizes)*	Unit of Address Resolution	Ženklo ilgis
1978 *(1980)*	Intel 8086 *(w/Intel 8087)*	16 b	½w, w, 2d *(w, 2w, 4w)*	— *(2w, 4w, 5w, 17d)*	½w, w, ... 7w	8 b	8 b
1989	Intel 80486 (x86)	16 b	½w, w, 2d, w, 2w, 4w	2w, 4w, 5w, 17d	½w, w, ... 7w	8 b	8 b
2010	x86-128	16 b*8 (dqword)	½w, w, dw, qw, dqw	dw, qw, 5w, 17d, dqw	½w, w, ... 7w	8 b	8 b
2010	IA-128	128 b	8 b, 16 b, ¼w, ½w, w	¼w, ½w, w	41 b	8 b	8 b

Metai	Kompiuterių architektūra		Fizinio adresavimo erdvė	Žodžio ilgis *w*	Susijusios failų sistemos
1978	x86-16 *8086+*		16 bitų	16 b	FAT-16
1989	IA-32 *(x86-32)*	*i486+*	32 bitų	16 b*2 (dword)	FAT-32, NTFS ext3
1999		*Pentium III+*	36 bitų *(PAE)* *(32 bitų fizinė)*		
2001	IA-64 *Itanium*		64 bitų	16 b*4 (qword)	NTFS, exFAT, ext4
2004	x86-64 *Intel 64, AMD 64*			64 b	
2010	x86-128 *Sandy Bridge, Bulldozer*		128 bitų	16 b*8 (dqword)	ZFS, GPFS *ribotai:* NTFS, exFAT, ext4
2014	IA-128 *Kittson*		128 bitų	128 b	

RM darbo režimai:

Long mode - 64 mode
Long mode – extra 128 mode
Long mode - Protected 32 Mode
Compatibity IA-32 mode

VM darbo režimai:

Long mode: 128-bit mode (128 bitų + AVX 256 bitų)
Long mode: 64-bit mode (dalinis funkcionalumas – 64 bitų)
Long mode: Compatibility mode (labai ribotas funkcionalumas – 32/16 bitų)

Kiekvienam iš režimu yra atitinkamai galimybė dirbti vartotojo arba supervizoriaus režimu.

RM ir VM naudojami matavimo vienetai

SI vienetų sistemos:

Pavadinimas	**Priesaga** *(SI)*	**Trumpinys**	**Apytikslis dydis baitais** *(IEC)*	**Dydis baitais** *(SI)*	*Dydis dešimtainiu formatu*	**Pavadinimas dešimtainiu formatu**	**Tarptautinis pavadinimas angl.**
bitas *(bit)*	--	1 b	1/8 B	1/8 B	1/8	--	--
pusbaitis *(nibble)*	---	½ B	1/2 B	1/2 B	1/2	---	---
baitas *(byte)*	--	1 B	1,00 B	1 B	10^0	vienas	*one*
Kilobaitas	kilo	1 kB	0,98 KiB	1000 B	10^3	tūkstantis	*thousand*
Megabaitas	Mega	1 MB	0,95 MiB	1000 kB	10^6	milijonas	*million*
Gigabaitas	Giga	1 GB	0,93 GiB	1000 MB	10^9	milijardas	*billion*
Terabaitas	Tera	1 TB	0,91 TiB	1000 GB	10^{12}	trilijonas	*trillion*
Petabaitas	Peta	1 PB	0,89 PiB	1000 TB	10^{15}	kvadrilijonas	*quadrillion*
Eksabaitas	Exa	1 EB	0,87 EiB	1000 PB	10^{18}	kvintilijonas	*quintillion*
Zetabaitas	Zetta	1 ZB	0,85 ZiB	1000 EB	10^{21}	sikstilijonas	*sextillion*
Jotabaitas	Yotta	1 YB	0,83 YiB	1000 ZB	10^{24}	septilijonas	*septillion*
Brontobaitas	*Bronto*	*1 BB*	0,81 *BiB*	1000 YB	10^{27}	oktilijonas	*octillion*
Ksonabaitas	Xona	1 XB	0,81 XiB	1000 YB	10^{27}	oktilijonas	*octillion*
Vekabaitas	Weka	1 WB	0,79 WiB	1000 XB	10^{30}	naintilijonas	*nonillion*
Vundabaitas	Vunda	1 VB	0,77 ViB	1000 WB	10^{33}	decilijonas	*decillion*
Udabaitas	Uda	1 UB	0,75 UiB	1000 UB	10^{36}	undecilijonas	*undecillion*

IEC vienetų sistema:

Architektūra	**Pavadinimas**	**Priesaga** *(IEC)*	**Trumpinys**	**Dydis dvejetainiu formatu**	**Apytikslis dydis baitais** *(SI)*	**Dydis baitais** *(IEC)*	*Apytikslis dydis dešimtainiu formatu*	*Dydžių grupės pavadinimas*
32 b	Kibibaitas	Kibi	1 KiB	2^{10}	1,02 kB	1024 B	$1{,}024*10^3$	tūkstančiai
	Mebibaitas	Mebi	1 MiB	2^{20}	1,05 MB	1024 kiB	$1{,}049*10^6$	milijonai
	Gibibaitas	Gibi	1 GiB	2^{30}	1,07 GB	1024 MiB	$1{,}074*10^9$	milijardai
64 b	Tebibaitas	Tebi	1 TiB	2^{40}	1,10 TB	1024 GiB	$1{,}100*10^{12}$	trilijonai
	Pebibaitas	Pebi	1 PiB	2^{50}	1,17 PB	1024 TiB	$1{,}126*10^{15}$	kvadrilijonai
	Eksbibaitas	Exbi	1 EiB	2^{60}	1,15 EB	1024 PiB	$1{,}153*10^{18}$	kvintilijonai
128 BITŲ	Zebibaitas	Zebi	1 ZiB	2^{70}	1,18 ZB	1024 EiB	$1{,}181*10^{21}$	sikstilijonai
	Jobibaitas	Yobi	1 YiB	2^{80}	1,21 YB	1024 ZiB	$1{,}209*10^{24}$	septilijonai
	Bronbibaitas	*BronBi*	*1 BiB*	2^{90}	1,24 *BB*	1024 YiB	$1{,}238*10^{27}$	oktilijonai
	Ksobibaitas	Xobi	1 XiB	2^{90}	1,24 XB	1024 YiB	$1{,}238*10^{27}$	oktilijonai
	Webibaitas	Webi	1 WiB	2^{100}	1,27 WB	1024 XiB	$*1{,}268*10^{30}$	naintilijonai
	VubiBaitas	Vubi	1ViB	2^{110}	1,30 VB	1024 WiB	$*1{,}298*10^{33}$	decilijonai
	UbiBaitas	Ubi	1UiB	2^{120}	1,33 UB	1024 ViB	$*1{,}329*10^{36}$	undecilijonai

Kiti siūlyti dydžiai:
Yottabyte(YB) < *kiloYotta-byte(kYB)* <*MegaYotta-byte(MYB)* <*GigaYotta-byte(GYB), TeraYotta-byte(TYB), PetaYotta-byte(PYB)*
Yottabyte(YB) < *Brontobyte(BB)* < *Geopbyte* < *Saganbyte* < *Pijabyte* < *Alphabyte* < *Kryatbyte*

RM naudojamos failų sistemos

Pagrindinės failų sistemos:

Prist. Metai	Failų sistema	Pilnas pavad.	Adreso ilgis	Maks. failo dydis	Maks. skirsnio dydis	Klast-erio dydis	Maks. failo dydis pagal klasterio dydį						Maks. skirsnio dydis pagal klasterio dydį					
							4KiB	8KiB	16KiB	32KiB	64KiB	128KiB	4KiB	8KiB	16KiB	32KiB	64KiB	128KiB
1987	FAT-16	**F**ile **A**llocation **T**able	16-bitų	4 GiB	4 GiB	iki 64 kiB	4GB					---	256 MiB	512 MiB	1 GiB	2 GiB	4 GiB	---
1996	FAT-32		32-bitų	4 GiB	2 TiB	iki 32 KiB	4GB				---	---	8 GiB	16 GB	32 GiB	2 TiB	---	---
1993	NTFS 6.0	**N**e**t**work **F**ile **S**ystem	64-bitų	256 TiB	256 TiB	iki 64 KiB	16	32	64	128	256 TiB	---	16 TiB	32 TiB	64 TiB	128 TiB	256 TiB	---
2001	exFAT *FAT-64*	**Ex**tended **F**ile **A**llocation **T**able	64-bitų	16 ZiB *(32MiB cluster)*	64 ZiB *(32MiB cluster)*	iki 32 MiB	256 MiB	---	--	32 GiB	---	256 TiB	256 MiB	---	--	32GiB	--	256TiB
2004	ZFS	**Z**ettabyte **F**ile **S**ystem	128-bitų	16EiB	16EiB	N/A	?	?	?	?	16EiB	?	?	?	?	?	16EiB	256UiB (2^{128}B)
2001	ext3	**3**-rd **Ext**ended File System	32-bitų	2TiB	32TiB	4KiB	2TiB	2TiB	---	---	---	---	16TiB	32TiB	---	---	---	---
2006	ext4	**4**-th **Ext**ended File System	48-bitų	16TiB (2^{44} B)	1EiB (2^{60}B)	Kintamas	---	---	---	---	---	---	---	---	---	---	--	--
			64-bitų	1EiB (2^{60} B)	1EiB (2^{60} B)													
2009	GPFS	**G**eneral **P**arallel **F**ile **S**ystem	64-bitų	512XiB (2^{99} B)	512XiB (2^{99} B)	4k nodes	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Bitiniai skirtumai tarp Giga ir Gibi:

Disko talpa	Reali talpa	Pristatytas
40 GB	37,26 GiB	2000-01
160 GB	149,01 GiB	2003-01
500 GB	465,66 GiB	2005-01
1 TB	0,91 TiB	2007-01
2 TB	1,82 TiB	2009-01
8 TB	7,28 TiB	2011-01

Disko talpa	Reali talpa	Pristatytas *(Moore'o progn.)*
40 TB	36,38 TiB	~2013
160 TB	145,52 TiB	~2015
500 TB	454,74 TiB	~2017
1 PB	0,89 PiB	~2019
2 PB	1,78 PiB	~2021
8 PB	7,28 PiB	~2023

OS Darbo režimai

Mūsų OS, kaip ir dauguma modernių Windows ar Linux OS, turi darbo režimus:

Vartotojo režimas *(user mode)* – norėdamas įvykdyti specializuotas funkcijas, "vartotojo lygio" programinės kodas privalo įvykdyti "system call" kvietimą, kur supervizoriaus režime OS įvykdys reikalaujamą užduotį ir gražins jos rezultatą atgal į vartotojo atmintį. Vartotojo lygmenyje dirba visos, operacinės sistemos leidžiamos("palaikomos", angl. "supported") taikomosios programos*("applications")*.

Supervizoriaus režimas *(supervisor mode)* – turi visas prieigos teises. Gali tiesiogiai su fizine*("hardware")* kompiuterine įranga, sisteminėmis magistralėmis, valdikliais*("controllers")*, koordinuoti teisingą tvarkyklių*("drivers")* darbą.

Taip pat yra dar vienas, tačiau mūsų OS nerealizuojamas režimas:

Hypervizoriaus režimas *(hypervisor mode)* – virtualizacijos lygmuo, leidžia paleisti kelias operacines sistemas x86 architektūroje vienu metu(x86 virtualizacija). Šią technologija įgyvendina Core 2 Duo E6300 ir naujesni, bei AMD64 ir naujesni, centriniai procesoriai.
x86 virtualizacija dar vadinamas aparatūrine virtualizacija*(hardware virtualization)*.
Hypervizorius dar vadinamas *"virtualios mašinos monitoriumi"(VMM)*.

x86-128 procesoriaus darbo režimai

IA-32, Normalus režimas:

1) ***Apsaugotas režimas*** *(„protected mode" arba „protected virtual address mode")* – šio režimo esmė, naujesnių procesorių instrukcijas pradėti vykdyti "realiuojo režimu"(real-mode), taip išsaugant senesnių procesorių funkcionalumo supratimą naujesnėje įrangoje. Į apsaugotą režimą įmanoma patekti tik tuomet, kaip sisteminė programinė įranga reikiamai užpildo keletą deskriptoriaus lentelių, bei **CR0** procesoriaus registre, apsaugos bitui nustato **PE=1**. *(PE – Protection Enable)*.

Mūsų operacinės sistemos atveju, veikiant šiam režimui, mes emuliuosime 16 bitų instrukcijas esant 64 arba 128 bitų procesoriui. Kalbant apie registrus, šis režimas dažnai reiškia tiesiog „vyresniųjų N-bitų ignoravimą", taip traktuojant jaunesniuosius N-bitų kaip pilną registro reikšmę ar adresą.

2) ***realus režimas*** *(„real mode")* – šiame režime neegzistuoja jokia atminties apsauga, neegzistuoja multitasking'as, neegzistuoja privilegijų lygiai. Charakterizuotas 20-ies bitų segmentuotoje atminties adresavimo erdvėje (1 MiB dydžio). Yra neribojamas tiesioginis programinės įrangos prieinamumas prie atminties, aparatūrinės įrangos bei I/O įrenginių.

IA-32e, *(„e"-* ***Extended)*** ilgas režimas („long mode"). Šis režimas yra skirstomas į:

1) ***palaikomas režimas*** („compatibility mode") - šio režimo pagalba, neįvykdę programos pakartotinio kompiliavimo kitoje terpėje, galime vykdyti 16 bei 32 bitų programas. PAE*(„Physical address extension")* pagalba, programinė įranga gali pasiekti operatyviosios atminties vietas už 4GiB ribos.
2) ***Ilgas-64 bitų režimas*** („long 64-bit mode") – yra originalus 64 bitų procesorių, esant 64 bitų operacinei sistemai ir 64 bitų taikomajai programai, darbo režimas. Esant 128 bitų procesoriui ir 128 bitų operacinei sistemai, esant šiam režimui, galime vykdyti 64 bitų programas.
3) ***Ilgas-128 bitų režimas*** („long 128-bit mode") – yra originalus 128 bitų procesorių, esant 128 bitų operacinei saistei ir 128 bitų taikomąjai programai, darbo režimas. 64 bitų procesoriuose šis režimas – nėra palaikomas. Šis režimas dar kartais vadinamas IA-32ee (ee – „extra extended").

Branduolio tipas

Terminai:

***IPC** (Inter-Process Communication)* – tai keitimasis duomenimis tarp kelių gijų viename arba keliuose procesuose. Metodai: failinis, semaforų, signalų, "shared" atmintimi ir kt.

Operacinė sistema gali būti paremta vienu iš 3 branduolio variantu:

Hibridinis branduolys*(populiariausias, juo paremta mūsų OS)*:

Populiariausios OS: Windows NT sistemos (NT 4, 2000, XP, Server 2003/2008, Vista, 7), BeOS, Mac OS-X, SUSE.

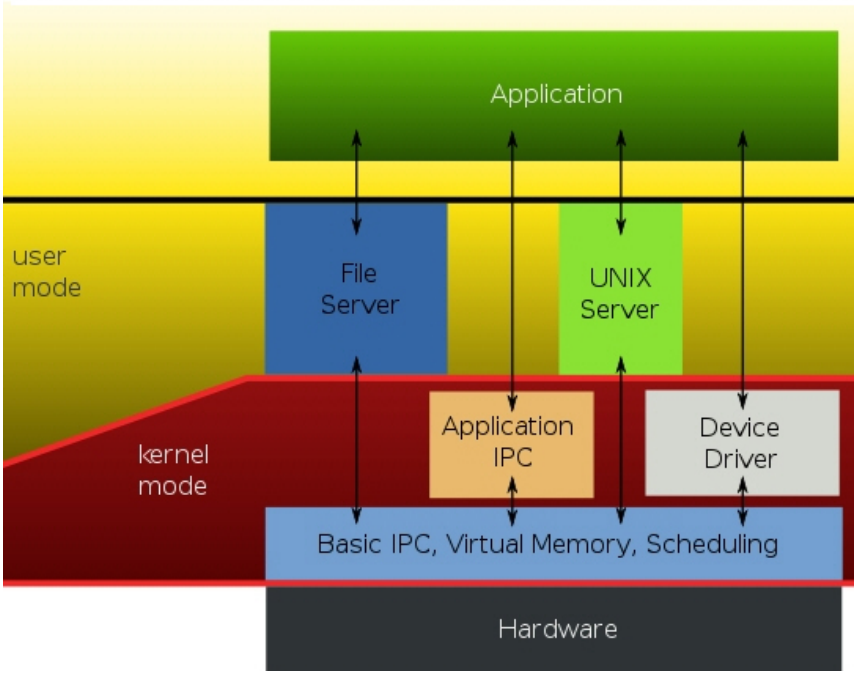
Principai:

Supervizoriaus režime vyksta bazinis komunikavimas, aptarnaujama virtuali atmintis, dirba įrenginių tvarkyklės(angl. "drivers"), vyksta komunikacija tarp taikomųjų programų(angl. "applications")

Vartotojo režime, skirtingai nei monolitinio branduolio atveju, yra leidžiama tiesiogiai*(nereikalaujant supervizoriaus režimo)* dirbti su failų serveriu*[failų sistema]*(Windows sistemoje tai vadinama "katalogų sistema") ir UNIX procesų serverį(notacijos "fork()", "exec()", "wait()" ir "exec()") *[Windows sistemoje procesų serverį atitinka panašus analogas "task manager"]*.

Svarbi pastaba: perėjus nuo monolitinio prie hibridinio branduolio varianto, buvo žymiai sumažintas perjungimų tarp user/supervisor režimų skaičius per sekundę*(pvz. Windows Me -> Windows XP atveju, nuo 3500k/s iki 150k/s kai yra dirbama to paties pajėgumo procesoriumi)*, o tai, be abejonės, leidžia žymiai padidinti sistemos našumą.

"Hibridinio branduolio"
angl. "Hybrid kernel")
pagrindu paremtos OS



Sistemos dalis

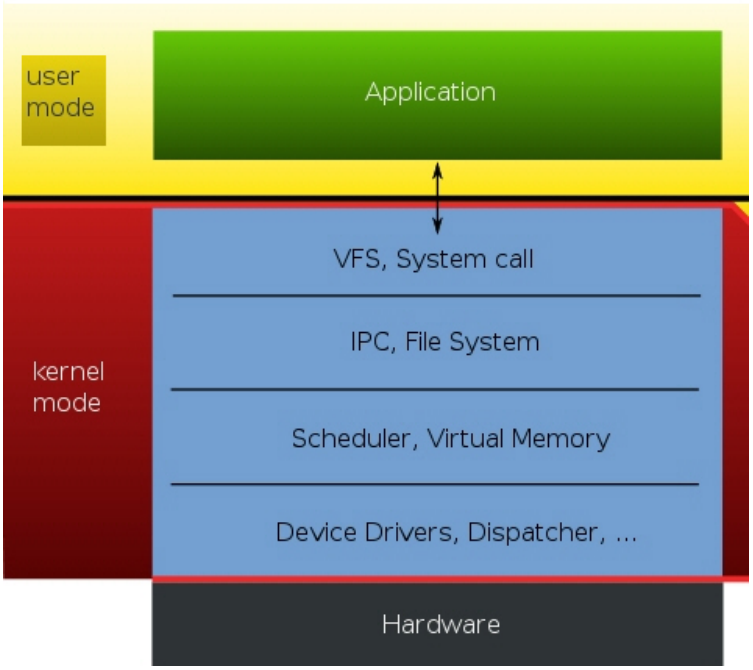
Operacinė sistemos dalis

Kiti, kartais naudojami, OS branduolių variantai(mūsų sistemoje neaptariami):

Monolitinis branduolys:

Populiariausios OS: MS-DOS, Windows: 95, 98, Me, Mac-OS 8.6, SunOS4, OpenSolaris, FreeBSD.

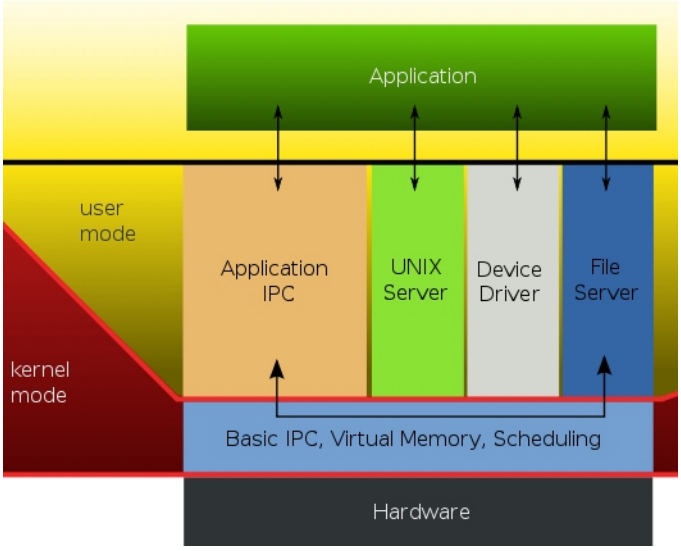
Molonitinio branduolio
(angl. Monolithic Kernel)
pagrindu paremtos OS



Microbranduolys:

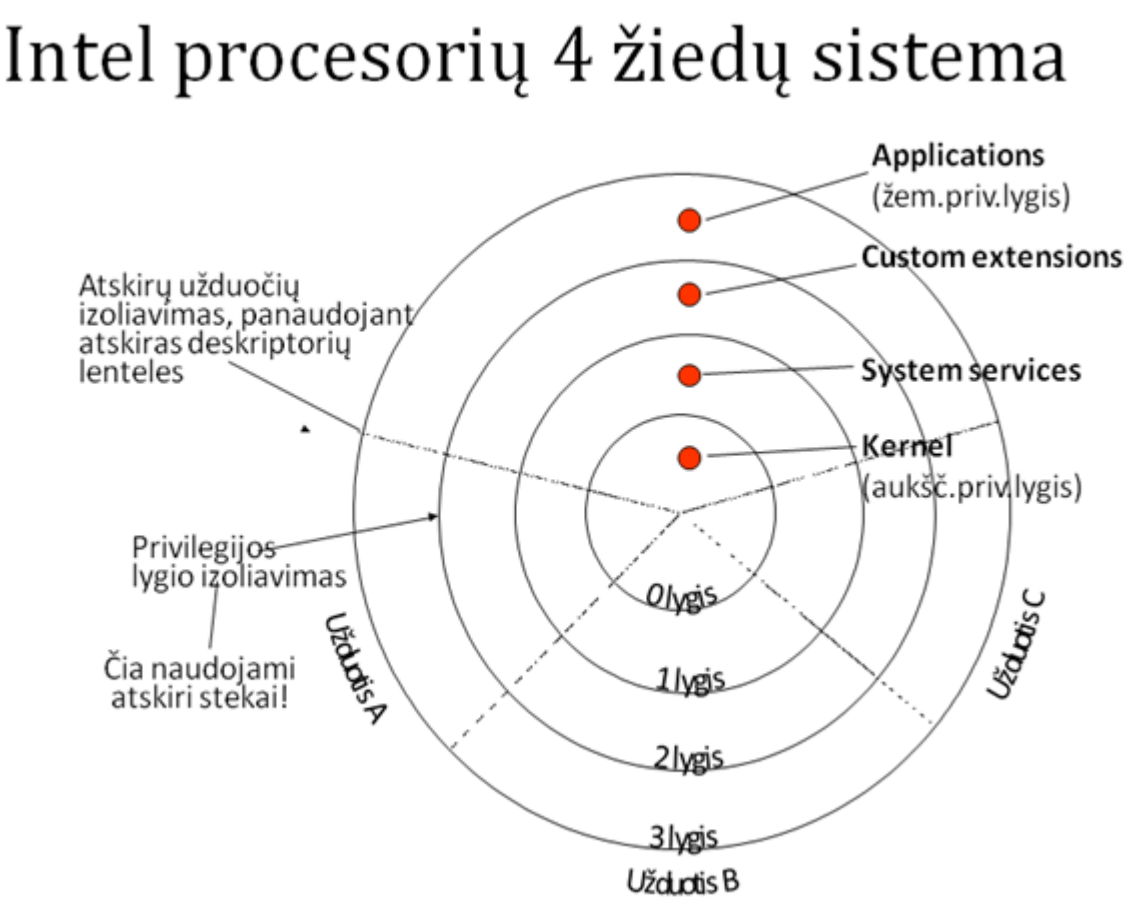
Anksčiau plačiau plėtojo Unix BSD. Šiuo metu tai yra retai naudojama technologija operacinėse sistemose, kadangi bet kokio serviso gavimas/aptarnavimas sistemos atžvilgiu yra nepalyginamai "brangesnis" (reikalaujantis daugiau laiko ir resursų) nei mololitinio ar hibridinio branduolio atveju.

MIKROBRANDUOLIO *(angl. Microkernel)*
pagrindu paremtos OS



Atminties apsauga

1.Izoliacijos lygiai:



2.NX *(No eXecute)* bitas. Taip vadinamas 63-bitas*(arba 127-as bitas)* puslapiavimo lentelėje, nurodantis ar toje eilutė pateikiamas adresas yra nuoroda į vykdomąją komandą ar ne.

Mūsų operacinė sistema, kaip ir modernios Windows OS, naudoja 2 iš 4 apsaugos lygių:

0 žiedas (Ring 0) – esant šiam apsaugos lygiui procesorius dirba supervizoriaus*(system)* režimu.

3 žiedas (Ring 3) – esant šiam apsaugos lygiui procesorius dirba vartotojo*(user)* režimu.

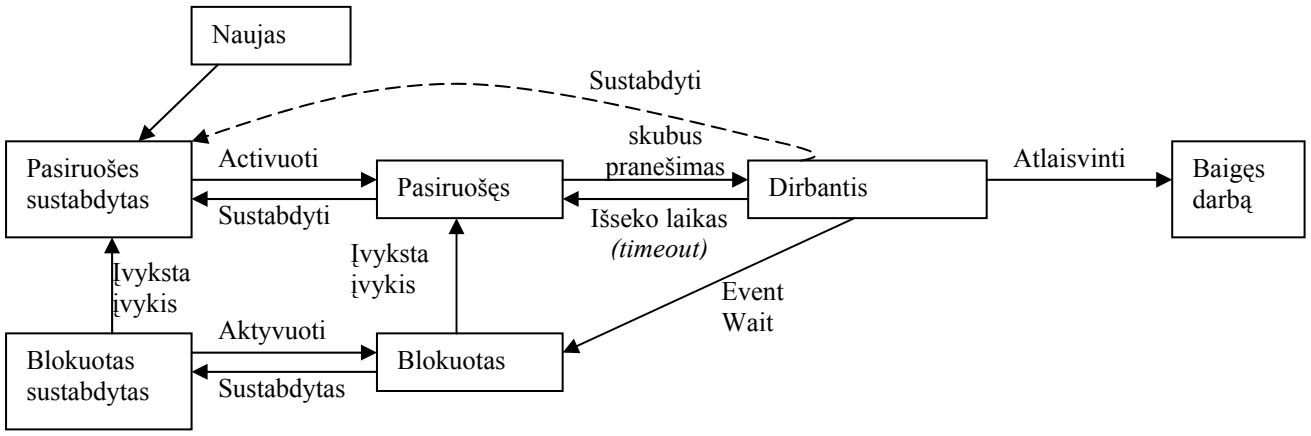
Pastaba. Dar yra(mūsų OS nerealizuojama):

0(-1) žiedas (fiktyvus 0-lygio žiedas, dar vadinamas “Ring -1”) – esant šiam apsaugos lygiui procesorius dirba hipervizoriaus*(hypervisor)* režimu. Šio žiedo nerealizuoja mūsų OS, tačiau realizuoja tokia OS sistema kaip Windows 7. Esant šiam izoliacijos lygiui sistema aplikacijai yra suteikusi visišką valdymo teisę, todėl kaip paprasta aplikacija yra virtualizuota Windows XP operacinė sistema veikiant Windows 7 OS.

OS hierarchija

Lygis		Vardas	Objektai	Pvz.
13	Vršutinis sluoksnis	Shell	Vartotojo darbo aplinka	Tai kalba. Yra sakiniai …
12		Vartotojo procesai	Vartotojo procesai	Quit, kill (pasiųsti signalą), suspend, resume (atstatyti)
11		Katalogai	Katalogais operuojame	Create, destroy, attach (prikabinti), detach (atkabinti), search, list
10		Įtaisai	Išoriniai įtaisai, pvz. printeris, monitorius	Sukurti, sunaikinti, atidaryti, uždaryti, skaityti (operuoja blokais arba simboliais)
9		Failų sistema	Failai	Create, destroy, open, close, read, write. (operuojame loginiais vienetais)
8		Komunikacijos	tuneliai (pipes)	Create, destroy, open, close, read, write. (iš kanalo gauname tai, ką jis duoda)
7	Vidurinis sluoksnis	Virtuali atmintis	Segmentai *(susiję su loginiais)*, puslapiai *(fizinis vienetas susijęs su diskais)*	Read, write, fetch (išrinkti)
6		Lokali antrinė atmintis	Duomenų blokai, įtaisų kanalai	Read, write, allocate(išskirti), free(atlaisvinti)
5		Primityvūs procesai	Primityvūs procesai, semoforai, įvairūs sąrašai	Suspend, resume, wait, signal (pasiųsti signalą)
4	Apatinis sluoksnis	Pertraukimai	Pertraukimų apdorojimo programos	Invoke (sukelti), mask, unmask, retry
3		Procedūros	Procedūros, kreipiniai į steką	Slėpti steką(“Mask stack”), kviesti(“call”), grąžinti(“return”)
2		Instrukcijų rinkinys	steko vykdymas, mikroprogramų interpretatorius, skaliariniai duomenys ir duom. masyvai(“arrays”)	Užkrauti (“load”), išsaugoti(“store”), add, substract, branch
1		Elektroninis	Registrai, vartai(gates), magristralės (“buses” FSB, USB)	Išvalyti(“clear”), persiųsti(“transfer”), aktyvuoti(“activate”), papildyti(“complement”)

1 proceso 7 būsenų diagrama:



Dirbantis – procesas turi procesorių.
Pasiruošęs – vienintelis trūkstamas resursas – procesorius.
Pasiruošęs sustabdytas – procesas turi resursą, tačiau jį blokavo kitas procesas arba jis buvo blokuotas ir gavo resursą.
Blokuotas – vykdant procesą jis negavo resurso.
Blokuotas sustabdytas – procesui esant blokuotam, einamasis procesas jį sustabdo.
Naujas – procesas sukurtas ir yra pastatomas į procesų būgną.
Baigęs darbą – procesas pašalinamas iš būgno.

Komandų formatai

Long mode: Compatibility mode(32/16-bit):

Pilna komandų sistema:

Long mode: 64-bit mode:

Legacy Prefixes	REX Prefix	Opcode	ModR/M	SIB	Displacement	Immediate
Grp 1, Grp 2, Grp 3, Grp 4 (optional)	(optional)	1-, 2-, or 3-byte opcode	1 byte (if required)	1 byte (if required)	Address displacement of 1, 2, or 4 bytes	Immediate data of 1, 2, or 4 bytes or none

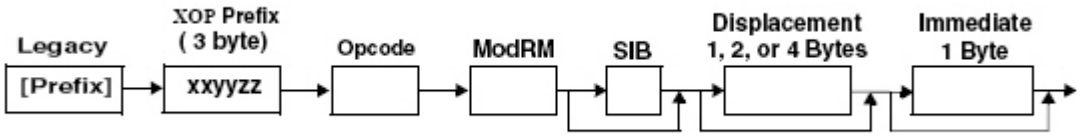
REX priesaga. 4 bitų ilgio. WYXZ formatas.

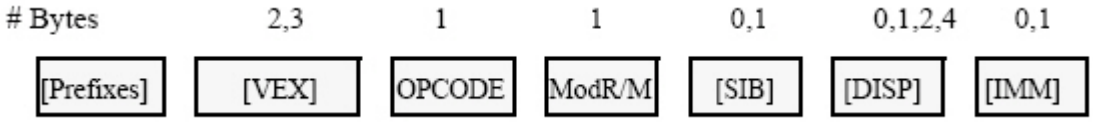
W bitas nurodo perėjimą nuo 32 bitų prie 64 bitų ilgio komandų(rex64), X, Y, Z bitai – tai nustatymų bitai.

Operacijos kodas(1, 2 arba 3 baitų operacijos kodas). Pirmasis baitas kartais traktuojamas kaip „escape prefix“.

Long mode: 128-bit mode:

Pilna komandų sistema:





[NEBŪTINA] Pasiliekanti priesaga(legacy prefix):
Tai operando-ilgio, adreso-ilgio arba segmento-ilgio perrašymo komanda, taip kartojimo (REP) arba magistralės užrakinimo priesaga(LOCK)
[NEBŪTINA] VEX priesaga

Operacijos kodas(1, 2 arba 3 baitų operacijos kodas). Pirmasis baitas kartais traktuojamas kaip „escape prefix“.

Puslapiavimo mechanizmas

Virtualiojo adreso transformacija į fizinį vadinama **adresų transliacija**.
Naujos kartos procesoriuose šiai transformacijai naudojamas:
Puslapiavimas – atmintis padalinama į fiksuoto dydžio blokus, vadinamus *puslapiais*. Uždaviniui spręsti reikalinga atminties sritis užima tam tikrą puslapių skaičių. Vienam uždaviniui išskirti puslapiai gali būti išdėstyti atmintyje bet kuria tvarka.

Senesniuose procesoriuose šiai transformacijai buvo naudojamas:
Segmentavimas – uždaviniui spręsti reikalinga atminties sritis vadinama segmentu.
Segmentas – ištisinė atminties sritis. Jo dydis priklauso nuo uždavinio poreikių.
Vienam uždaviniui gali būti suformuoti keli segmentai – kodo (programos), duomenų, steko.

Adresų transliacijai reikalingas papildomas žingsnis, kad jis būtų spartesnis, naudojami 2 būdai:

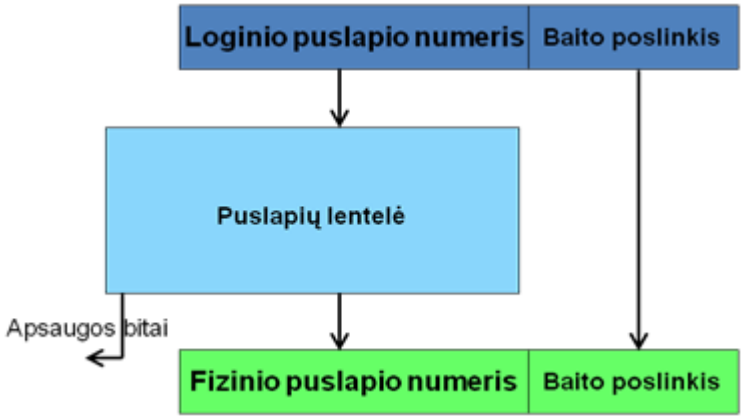
1.Puslapių lentelės dalį saugoti procesoriuje esančiame specialiame keše. Kiekvienas tokio kešo įrašas užtikrina greitą transliaciją, kreipiantis net į 1000 žodžių.

2.Papildoma priemonė – adresų transliaciją vykdyti lygiagrečiai su kreipiniu į kešą.

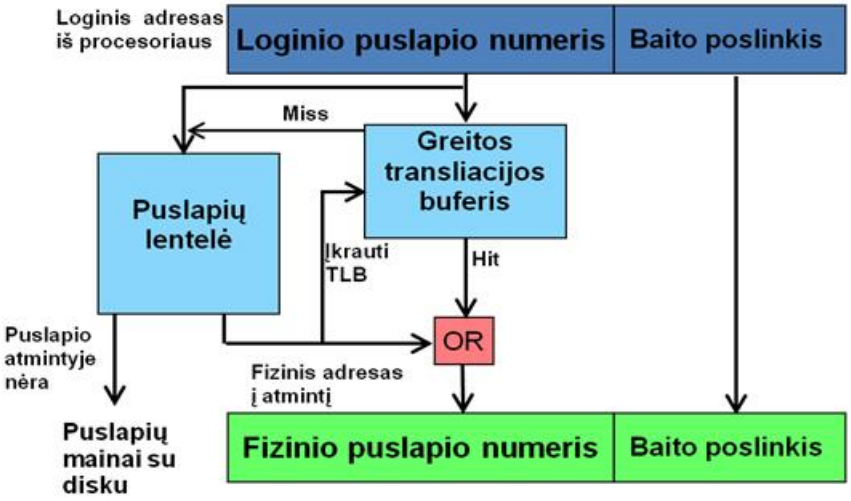
Santrumpos:

PML8 - page table level 8 page (page map level)
PML7 - page table level 7 page (page map level)
PML6 - page table level 6 page (page map level)
PML5 - page table level 5 page (page map level)
PML4 - page table level 4 page (page map level)
PDP – page table level 3 page (page directory pointer table)
PD – page table level 2 page (page directory)
PT – page table level 1 page (page table)
PML8E – page table level 8 entry (page map level entry)
PML7E – page table level 7 entry (page map level entry)
PML6E – page table level 6 entry (page map level entry)
PML5E – page table level 5 entry (page map level entry)
PML4E – page table level 4 entry (page map level entry)
PDPE/PDPTE – page table level 3 entry (page directory pointer table entry)
PDE – page table level 2 entry (page directory entry)
PTE – page table level 1 entry (page table entry)

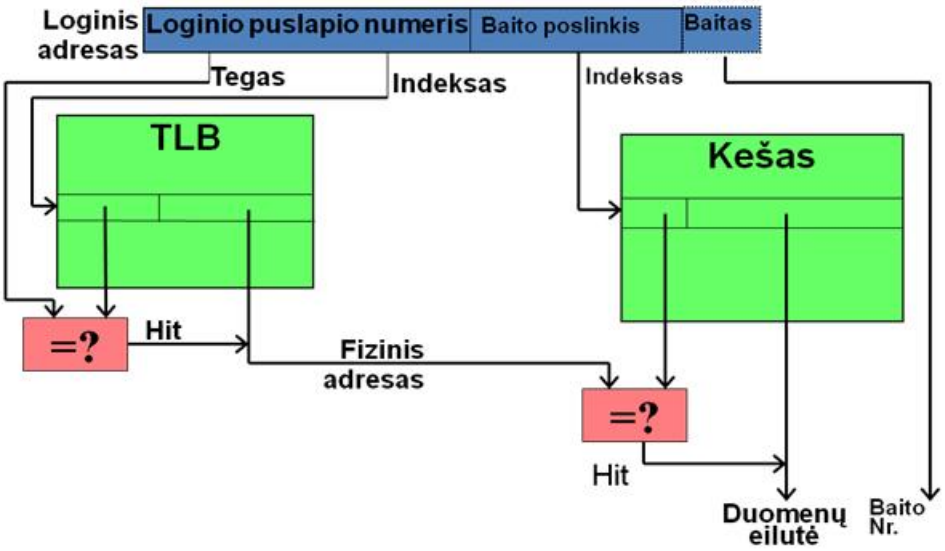
Primityvus puslapiavimo mechanizmas:



Greitos transliacijos buferis:

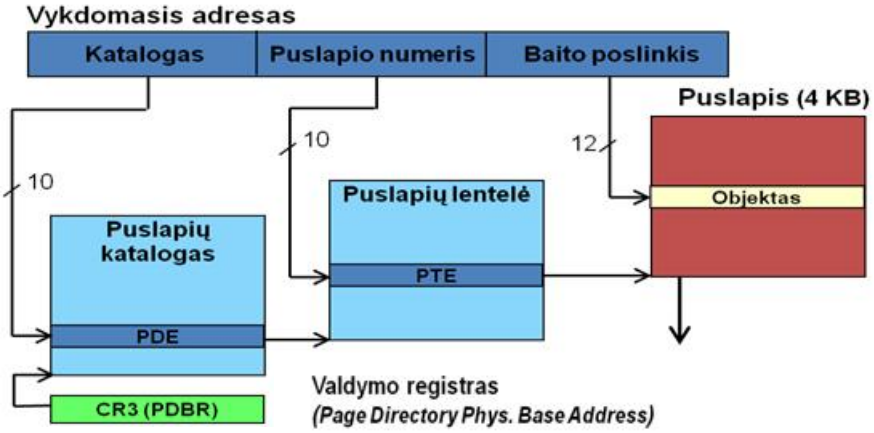


Tiesioginio atvaizdavimo TLB*(Translation lookaside buffer)* **ir kešas:**



IA-32 puslapiavimas*(32 bitų pagrindinis puslapiavimo mechanizmas)*:

[Tik pavyzdys, mūsų architektūroje palaikymas nenumatytas] *[transliacija į 4KiB puslapį]*



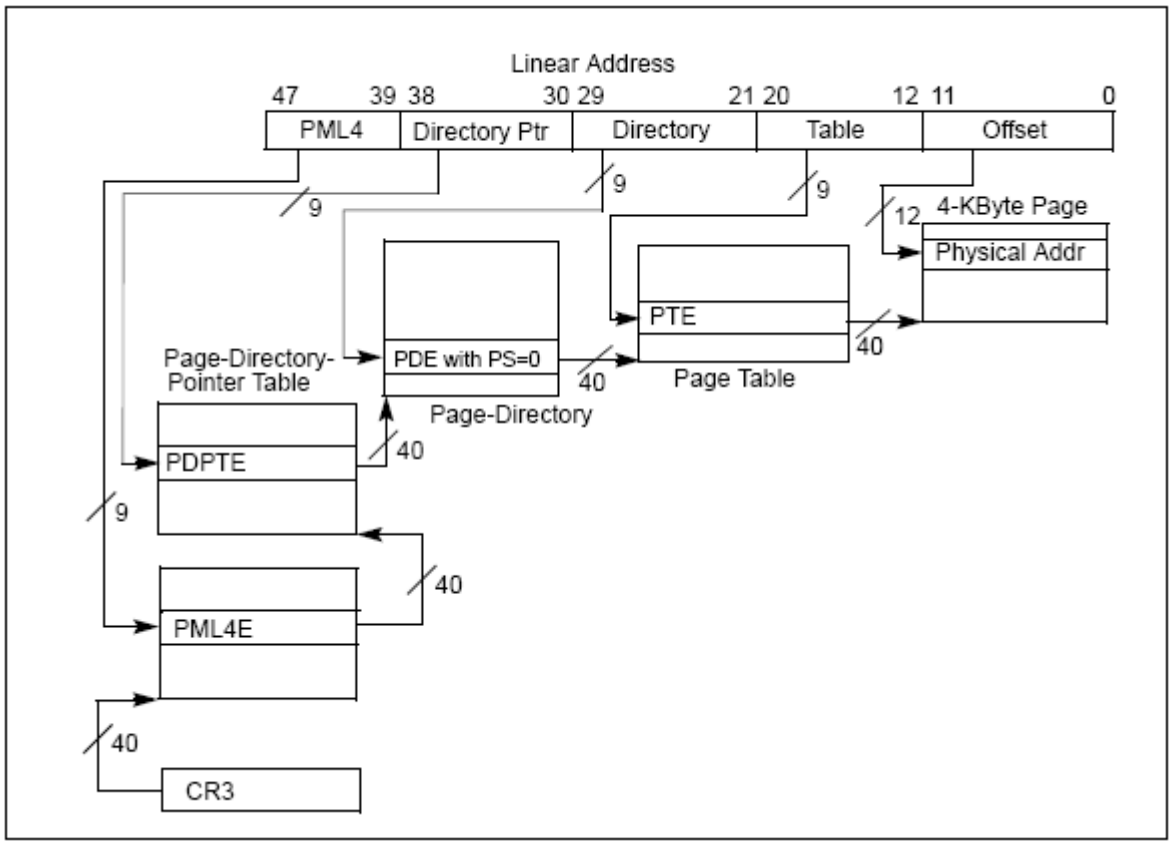
IA-32e puslapiavimas („long 64 mode“ režimu):

SVARBU: IA-32e puslapiavimo režime nebenaudojamas segmentacijos mechanizmas.

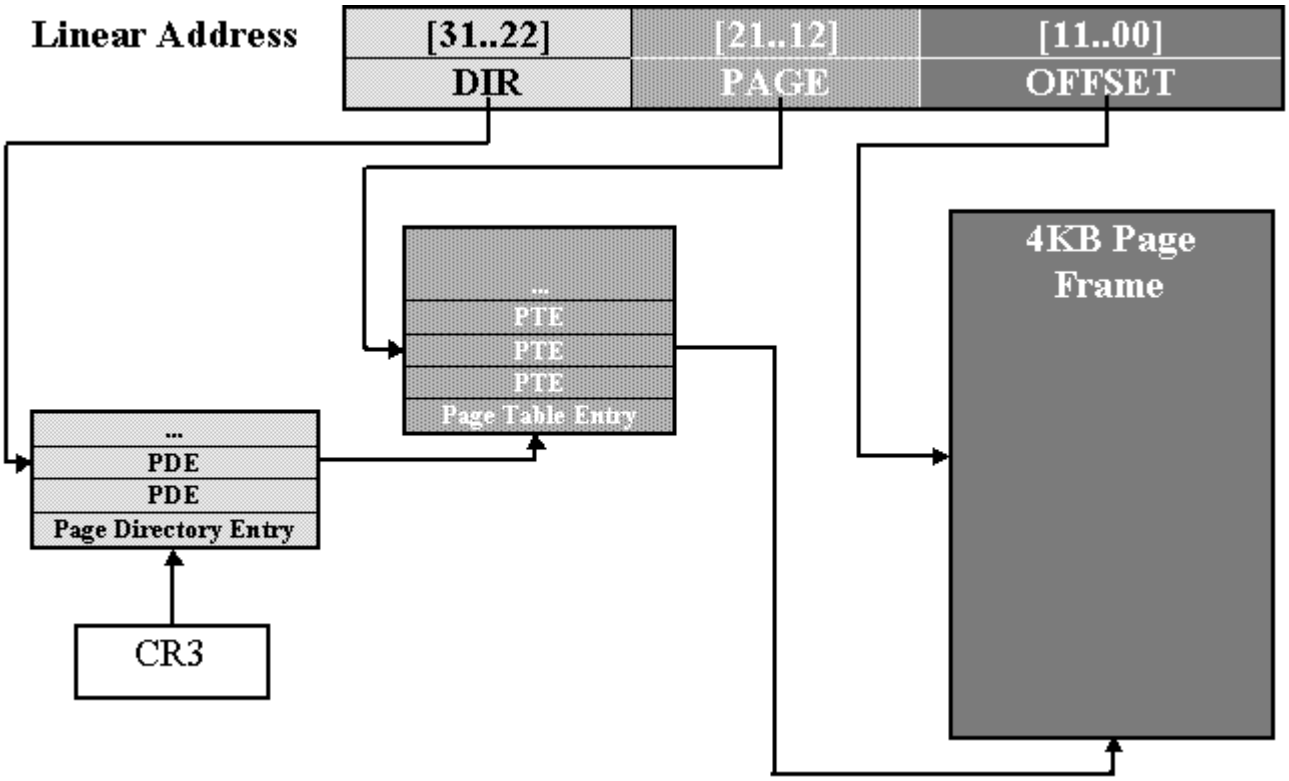
Procesoriui dirbant „long 64“ režime, ir naudojant IA-32e puslapiavimo būdą, yra naudojamas CR3 specialios paskirties registras. Pagal jo parametrus, yra vykdomas vienas iš 3 tiesioginio adreso transliavimo į puslapį būdų:

1.Transliavimas iš tiesioginio adreso į 4KiB lentelę.
2.Transliavimas iš tiesioginio adreso į 2MiB lentelę.
3.Transliavimas iš tiesioginio adreso į 1GiB lentelę.

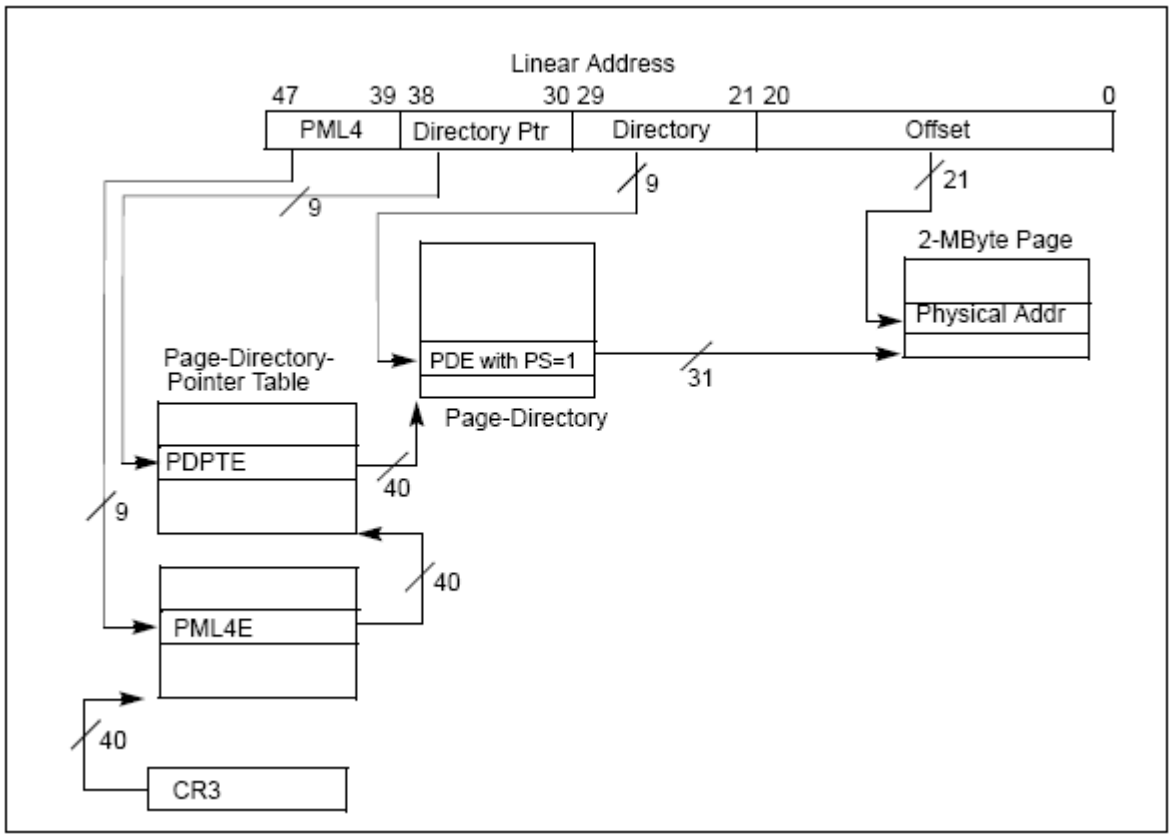
Žemiau pateiktos schemos kiekvienam iš atvejų bei CR3 specialaus registro turinys:



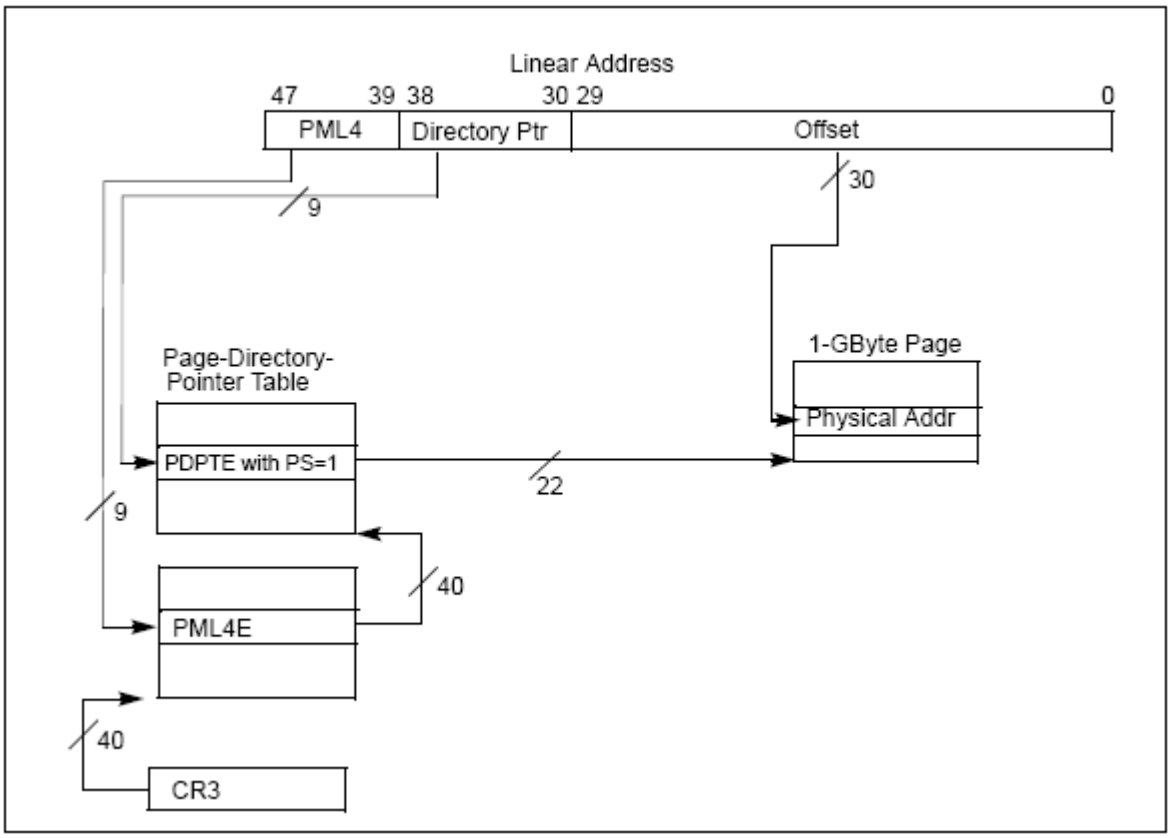
Tiesioginio*(linear)* adreso transliavimas į 4 KiB puslapį.



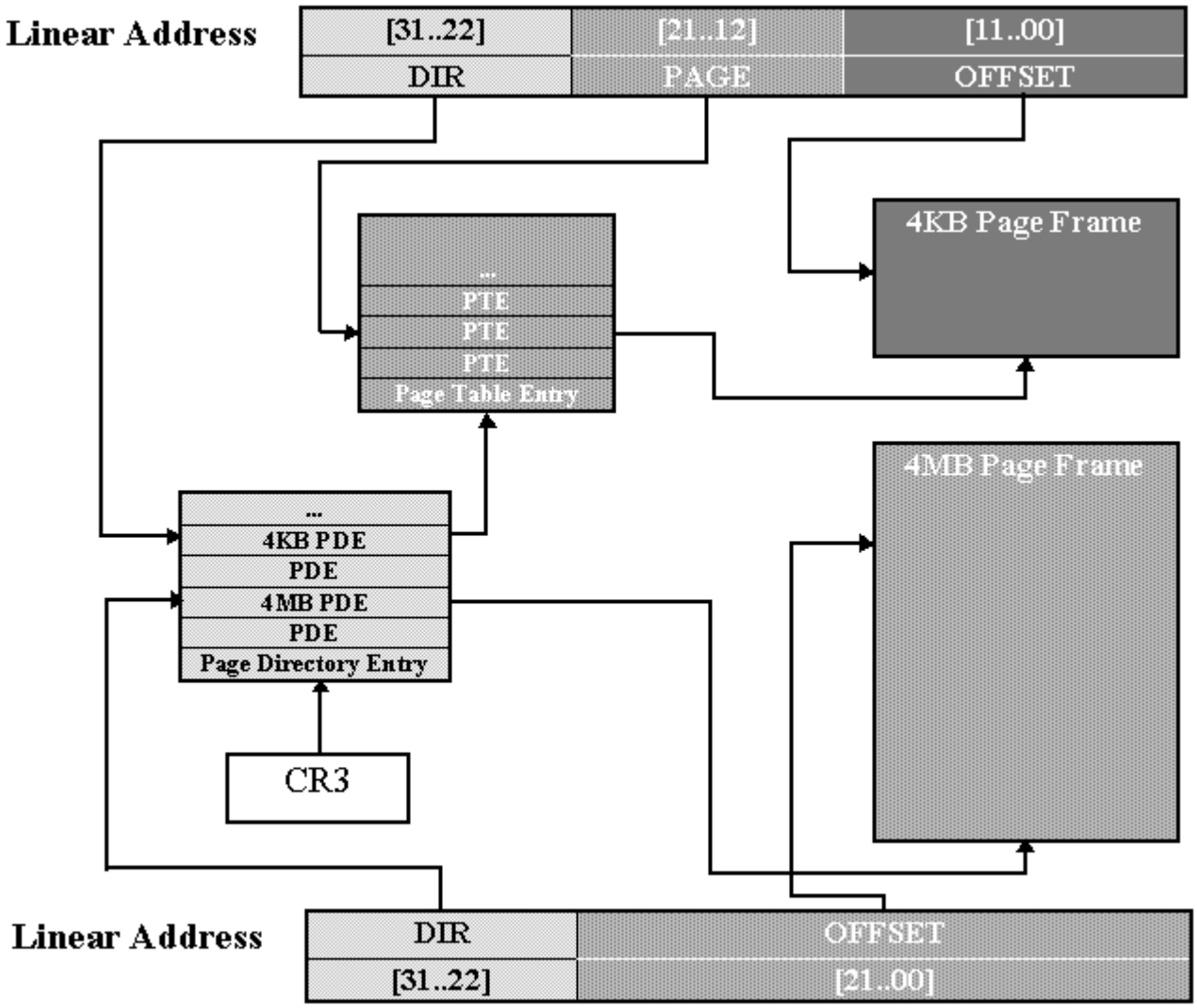
[PAVYZDYS iš SENESNIO CPU] Puslapio transliavimas 4 kiB puslapio dydžiui.



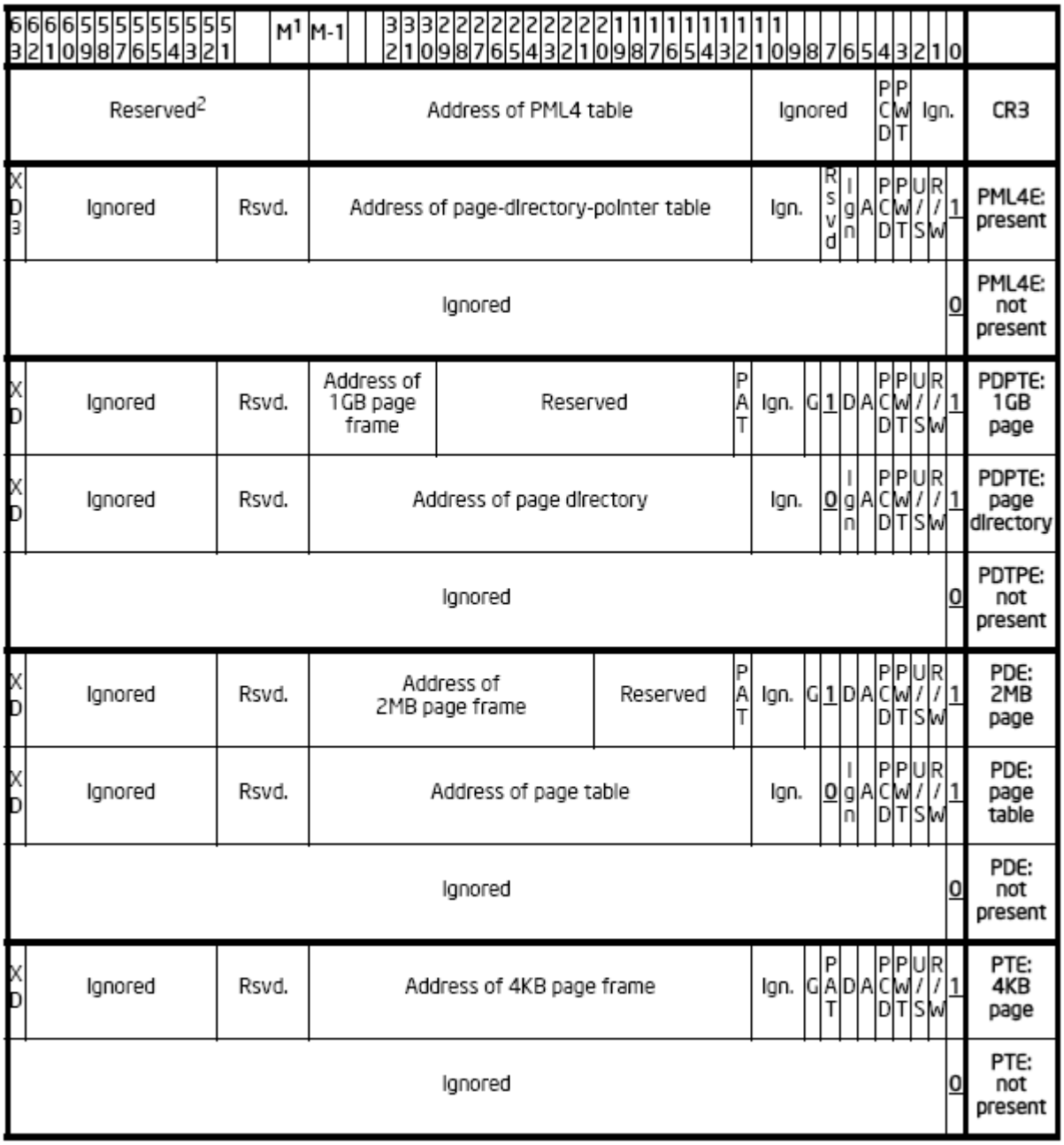
Tiesioginio*(linear)* adreso transliavimas į 2 MiB puslapį.



Tiesioginio*(linear)* adreso transliavimas į 1 GiB puslapį.



[PAVYZDYS iš SENESNIO CPU] Puslapio transliavimas į 4kiB ir 4MiB dydžio puslapius



CR3 spec. registro turinys

Pastabos dėl CR3 registro:

1. M yra registro bito pavadinimo „*MAXPHYADDR*“ trumpinys(Maksimalus adresuojamas fizinis adresas (bitų sk., įprastai – 32 bitai x86-32 ir x86-64 sistemose).
2. Rezervuoti laukeliai turi būti = 0.
3. Jeigu *IA32_EFER.NXE = 0* ir *P-flag*=1 (puslapiavimo strukūturos įrašas yra 1), *XD-flag* (63 bitas) yra rezervuotas.

*XD-flag kitaip dar yra žinomas kaip NX bitas.

IA-32ee puslapiavimas („extra long 128 mode“ režimu) [proto-futuristinis režimas]:

SVARBU: IA-32ee puslapiavimo režime nebenaudojamas segmentacijos mechanizmas.

Procesoriui dirbant „long 128“ režime, ir naudojant IA-32ee puslapiavimo būdą, yra naudojamas CR3 specialios paskirties registras. Pagal jo parametrus, yra vykdomas vienas iš 3 tiesioginio adreso transliavimo į puslapį būdų:

(x86-16/32)	1.Transliavimas iš tiesioginio adreso į 4KiB puslapių lentelę.	(max. 65 kiB / 4 GiB RAM)
(x86-32/64)	2.Transliavimas iš tiesioginio adreso į 2MiB puslapių lentelę.	(max. 4 GiB / 16 PiB RAM)
(x86-64/128)	3.Transliavimas iš tiesioginio adreso į 1GiB puslapių lentelę.	(max. 16 PiB / 256 UiB RAM)
(x86-128/teor.)	4.Transliavimas iš tiesioginio adreso į 512GiB puslapių lentelę.	(max. 256 UiB / 2^{256} B RAM)

Kiekvieno lygio daugiklis yra 512 (2^9), kadangi vienam išplėtimui yra skirti 9 bitai.

Apribojimai – 4KiB dydžio puslapio lentelės labiausiai taupo vietą, tačiau maksimalus jų adresuojamas fizinės atminties kiekis yra 4 GiB.

PTE, PDE, PDPTE, PML4, PML5, PML6, PML7, PML8 lentelių ir CR3 registro nustatymų bitai:

63 (127) – NX bitas (No eXecute) Jei 0(įvykdomas), kodas iš šio puslapio gali būti įvykdytas(tariama kad saugomos komandos/kodas), jeigu 1 – kodas negali būtį įvykdymas(tariama kad saugomi duomenys)

8 – jeigu =1, sindikuoja kad transliacija yra globali*(CR4.PGE=1)*, kitu atveju ignoruojama.

7 – PS/PAT(Paging Settings) bitas, nusako ar imti adresą iš puslapiavimo lentelės, ar imti adresą iš poslinkio(offset) virtualiame adrese.

Page size; 1 - this entry maps a 1-GByte page; 0 - this entry references a page directory

PAT – nurodo atminties tipą[4kiB paging/2MiB paging/1GiB paging], naudotą pasiekti 4kiB dydžio puslapiui pasiekti iš šio puslapiavimo lentelės įrašo.

6 – Dirty. Jeigu 1, nurodo kad programinė įranga(software) **rašė** į atmintį šio puslapio įrašu nurodytame atminties bloke(puslapyje).

5 – Access. Jeigu 1, nurodo kad programinė įranga(software) **skaitė** atmintį šio puslapio įrašu nurodytame atminties bloke(puslapyje).

4 – PCD (page-level cache disabled). Jeigu 0, į šią atminties vietą kreipsimės nenaudodami kešavimo.

3 – PWT(page-level write thought). Ar rašyti naudojant TBL(translation allocation buffer) ar ne.

2 – U/S(User/supervisor) bitas. Jeigu 0(***user***), tai į puslapiavimo lentelėje esantį įrašą kreiptis draudžiama vartotojo režimu(t.y. programa negali kreiptis). Priegą turi tik operacinė sistema dirbdama supervizoriaus režimu. Jeigu = 1, tai prieigą turi ir sistema ir vartotojas.

1 – R/W bitas. Jeigu 0, tai ta atminties vieta yra tik Read-only, t.y. galima tik nuskaityti. Jeigu 1 – galima ir rašyti.

0 – Present bitas. ar egzistuoja tokia vieta atmintyje, ar adresas fiktyvus. Jeigu 0 – fiktyvus, jeigu 1 – egzistuoja lentelė.

Transliacija į puslapį

4KiB puslapis:

IA-32 architektūroje su 2 lygių puslapių lentelėmis:

1024 eilutes po 4 baitus(32bitus)

IA-32e architektūroje su 4 lygių puslapių lentelėmis:

512 eilučių po 16 baitų (64bitus)

IA-32ee architektūroje su 4 lygių puslapių lentelėmis:

512 eilučių po 32 baitus(128bitus)

Transliacija į puslapį:

Paiimamas poslinkis IP, ir formuojamas Efektyvus adresas(kitaip dar vadinamas virtualiu adresu):
Jeigu tiesioginė adresacija yra suformuojama pagal adresavimo(MOD REG R/M) baitą, tai EA formuojamas:
1.panaudojant REX/VEX baitą, operacijos kodą,
2.pagal REX/VEX baitą yra suformuojamas:
SIB baitas(jeigu reikia)
3.Pagal SIB ir REX/VEX baitus, yra suformuojamas:
32bitų MOD REG R/M baitas
SUFORMUOJAME:
BE SIB baito:
OPK [REG] [R/M]
pvz.:
inc Rrrr Bbbb
<____ mov 1000.0100 ____>

Su SIB baitu:
OPK [REG] [SIB: INDEX] [SIB BASE]
pvz.:
inc Rrrr Xxxx Bbbb
<____ inc 1010.0100.1011 ____>

Taigi turime virtualų adresą.

Dabar formuojame absoliutų adresą panaudodami 2, 4 arba 8 lygių puslapiavimo lenteles.

Puslapiavimo lentelė randasi atmintyje.

JEIGU TURIME IA-32 (32 bitų CPU):

Puslapiavimo lentelę surandame paėmę vyriausius AKTYVIUS 10 bitų(t.y. 31b - 22b) iš virtualaus(EA) adreso, ir 20 bitų iš CR3 registro(31:22)

Taigi 20 bitų[ID#: #M-1(#31) - #12] iš CR3 adresas(kai 2 lygių puslapiavimas) yra fizinis adresas puslapiavimo lentelės atmintyje(DRAM) su pabaiga xxxxxxxxxxxxxxxxxxxxxxxxxxxx**000000000** b =
Ta pabaiga yra 10 bitų iš Virtualaus adreso(31-22 bitai)

JEIGU TURIME IA-32e (64 bitų CPU):

Jeigu būtų L5, tai būtų imami EA bitai: 56b - 48b
Jeigu būtų L6, tai būtų imami EA bitai: 63b - 57b [trumpesnis puslapis]

Puslapiavimo lentelę surandame paėmę vyriausius AKTYVIUS 9 bitus(t.y. 47b - 39b) iš virtualaus(EA) adreso, ir 40 bitų iš CR3 registro(51:12)

Taigi 40 bitų[ID#: #M-1(#51) - #12] iš CR3 adresas(kai 4 lygių puslapiavimas) yra fizinis adresas puslapiavimo lentelės atmintyje(DRAM) su pabaiga xxxxxxxxxxxxxxxxxxxxxxxxxxxx**000000000** b =
Ta pabaiga yra 9 bitai iš Virtualaus adreso(47-39 bitai)
bitai [2:0] yra 000

PML4 lentelė(64 bitų pločio įrašai):
12 info bitų [11-0]
[51-12] - nuoroda, kur ieškoti PDPT puslapio.
[63 bitas] - NX bitas
likę bitai nenaudojami

-
PDPTE adresas susideda iš:
0xYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYZZZZZZZZZ000
2:0 bitai visai nulis
11:3 bitai(z reikšmės) yra paimti iš (38-30 b) Virtualaus adreso
51:12 bitai yra paimti iš PML4 (51-12 b) turinio

JEIGU TURIME IA-32ee (128 bitų CPU):
Įrašo puslapyje ID - vyriausi aktyvūs 9 bitai(t.y. 83b - 75b) iš virtualaus(EA) adreso, ir 80 bitų iš CR3 registro.

Taigi 80 bitų[ID#: #M-1(#91) - #12] iš CR3 adresas(kai 8 lygių puslapiavimas) yra fizinis adresas puslapiavimo lentelės atmintyje(DRAM) su pabaiga[11:3] xxxxxxxxxxxxxxxxxxxxxxxxxxxxx000000000[000] b =
Ta pabaiga yra 9 bitai iš Virtualaus adreso(83-75 bitai)
bitai [2:0] yra 000

PML4 lentelė(128 bitų pločio įrašai)
12 info bitų [11-0]
[91-12] - nuoroda, kur ieškoti PDPT puslapio.
[127 bitas] - NX bitas
likę bitai nenaudojami

PDPTE adresas susideda iš:
0x[0000000000000000]YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYZZZZZZZZZ000
2:0 bitai visai nulis
11:3 bitai(z reikšmės) yra paimti iš (74-66 b) Virtualaus adreso
91:12 bitai yra paimti iš PML4 (91-12 b) turinio

Multiprograminė sistema

Galutinė operacinės sistemos versija galės dirbti su iki 32 programomis vienu metu.

Kiekvienai iš programų yra galimybė suteikti prioritetą nuo 1 iki 99.

Konkretus prioritetas atitinka taktų skaičių magistralėje. Esant maksimaliam – 99 prioritetui, kitoms programoms vykdyti yra skiriamas kas 100‘asis ciklas.

Vienas ciklas – tai:

1. Visų elementų poslinkis į kairę pusę per 1 elementą taktinėje magistralėje.
2. Vienos programinio kodo eilutės vienoje iš programų apdorojimas.
3. 1 kreipinys į „*resourceManager()* “ metodą – „*getResourceOrWait()*“.

Kiekviena nauja programa gauna atskirą vietą RAM atmintyje – t.y. jai priskiriamas atitinkamas puslapiavimo lygis.

Kokio lygio puslapiavimo lentelę programa gaus, nurodo programos pradžioje esanti direktyva:

```
.program [tiny/small/normal/big/huge/large/extra]
```

Jeigu yra naudojamas 4kiB ir 8 lygių puslapiavimas su 32 bitų RAM pločiu ir 128 bitų IA-32ee režimas:

Tiny – 1 pirmo lygio puslapis su 512 įrašų lentele į puslapius fizinėje atmintyje. Kiekviena eilutė adresuoja 4096 ram blokelius po 128 bitus. Vieną blokelį sudaro 8 fizinės ram eilutės, iš jų naudojamos pirmos 4, likusios 4 128 bitų sistemose nenaudojamos.* **~ iki 64MiB**

Small – 1antro lygio puslapis su 512 įrašų lentele(iki **512^2** 1 lygio puslapių eilučių) **~ iki 32GiB**

Normal – 1trečio lygio puslapis su 512 įrašų lentele(iki **512^3** 1 lygio puslapių eilučių) **~ iki 16TiB**

Big – 1ketvirto lygio puslapis su 512 įrašų lentele(iki **512^4** 1 lygio puslapių eilučių) **~ iki 8PiB**

Huge – 1šešto lygio puslapis su 512 įrašų lentele(iki **512^5** 1 lygio puslapių eilučių) **~ iki 4EiB**

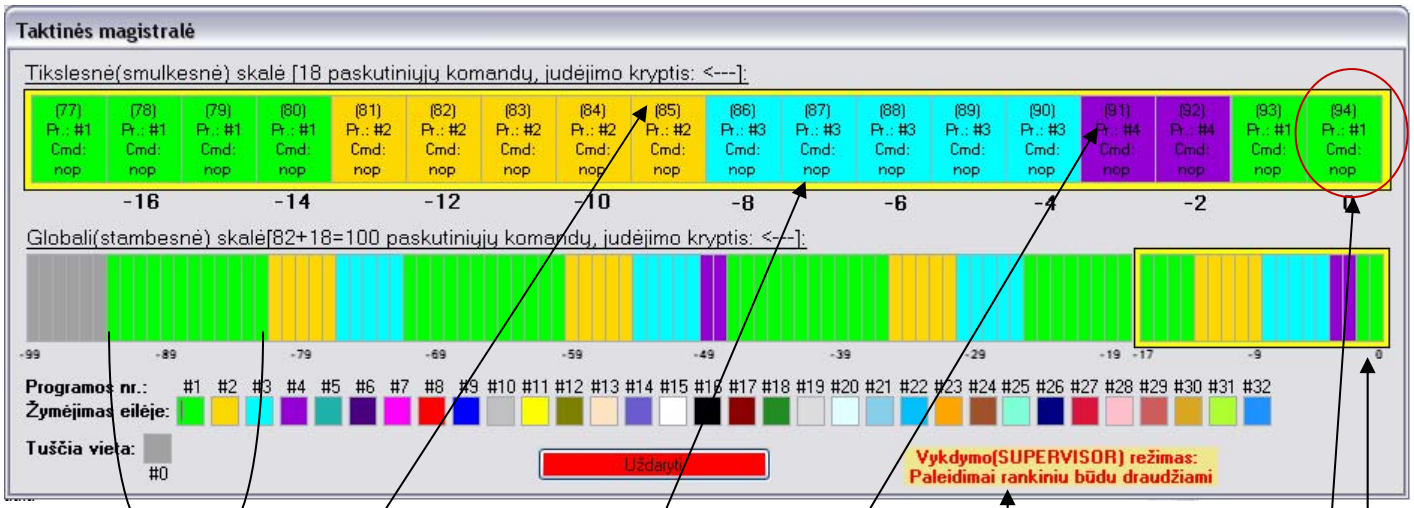
Large – 1šešto lygio puslapis su 512 įrašų lentele(iki **512^6** 1 lygio puslapių eilučių) **~iki 2ZiB**

ExtraLarge – 1septinto lygio puslapis su 512 įrašų lentele(iki **512^6** 1 lygio puslapių eilučių) **~ iki 1YiB**

Aštunto lygio(1 lentelė – **iki 512YiB**) puslapius valdo tik operacinė sistema.

Kiekviena programa saugo savo puslapiavimo lentelės adresą, kurį, ciklo metu įkrauna į CR3 registrą.

**Pastaba - IA-32e režime yra naudojamos pirmos dvi eilutes, o IA-32 režime skaidymo į blokelius nėra – t.y. vienas blokelis atitinka vieną fizinę ram eilutę.*



Takto numeris

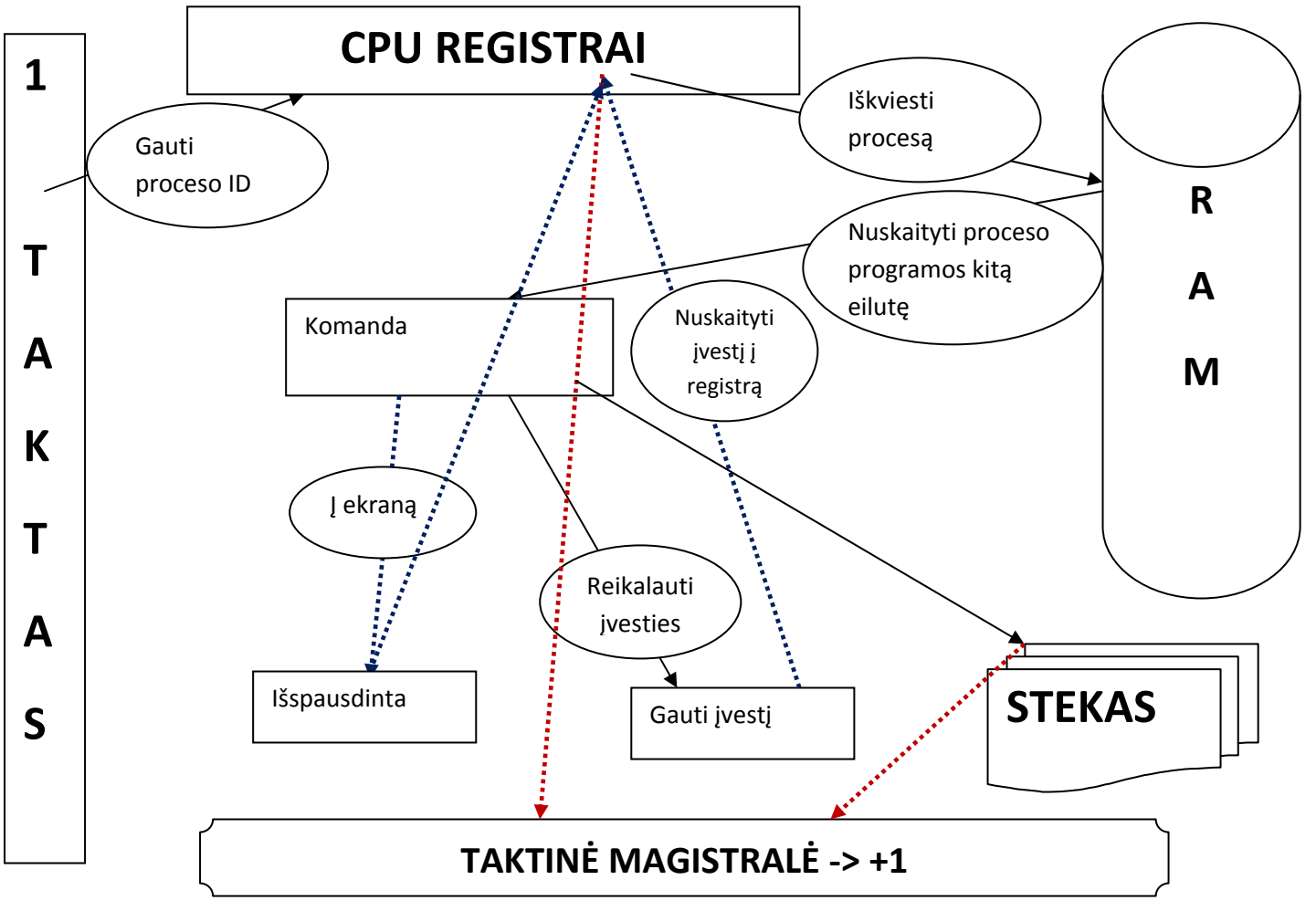
Įvykdyta komanda konkrečiame takte

Taktinis daugiklis = 12/99

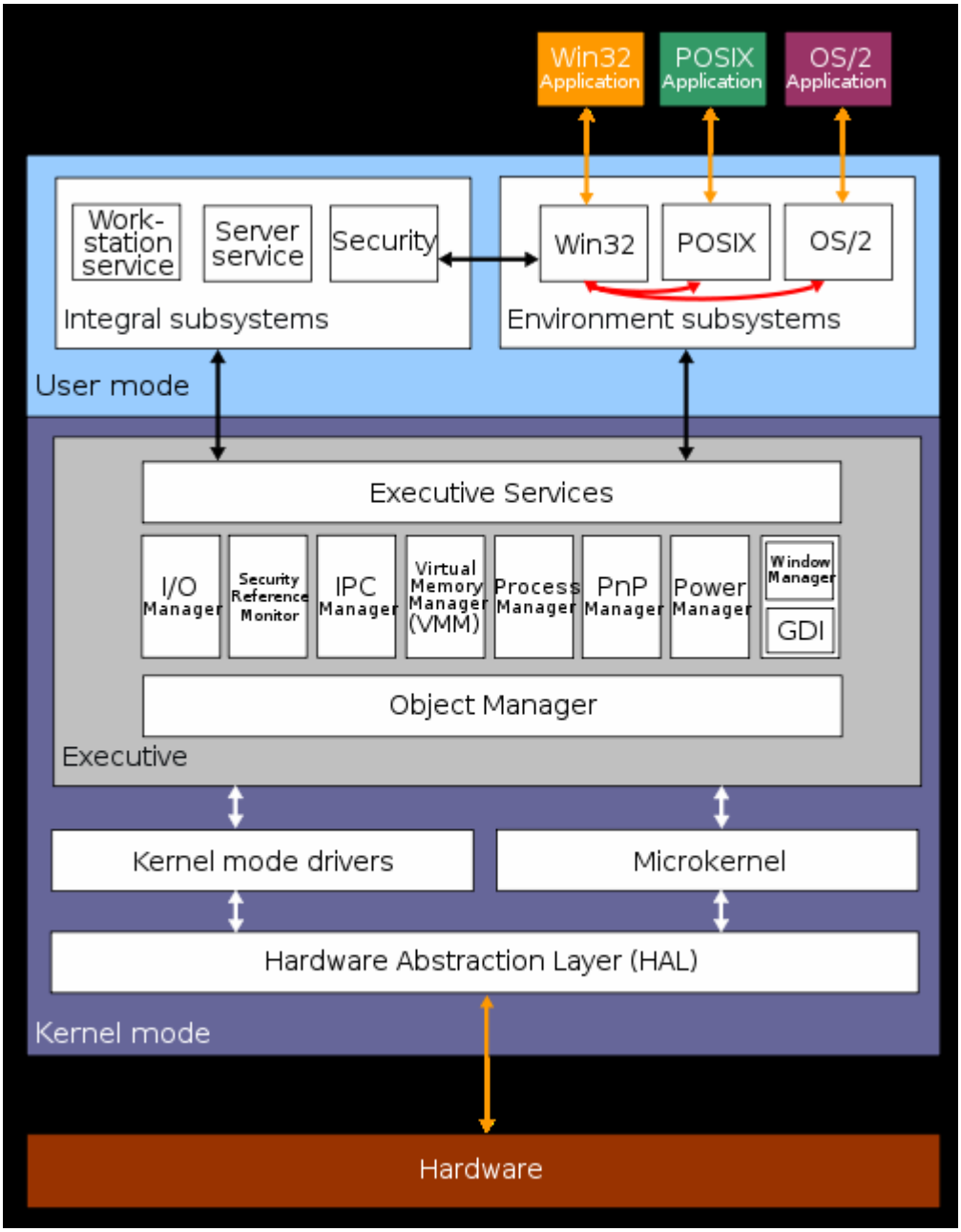
Programos numeris

Darbo režimas - supervizorius (vartotojo lygio pertraukimai neleidžiami)

Einamoji komanda (taktas)

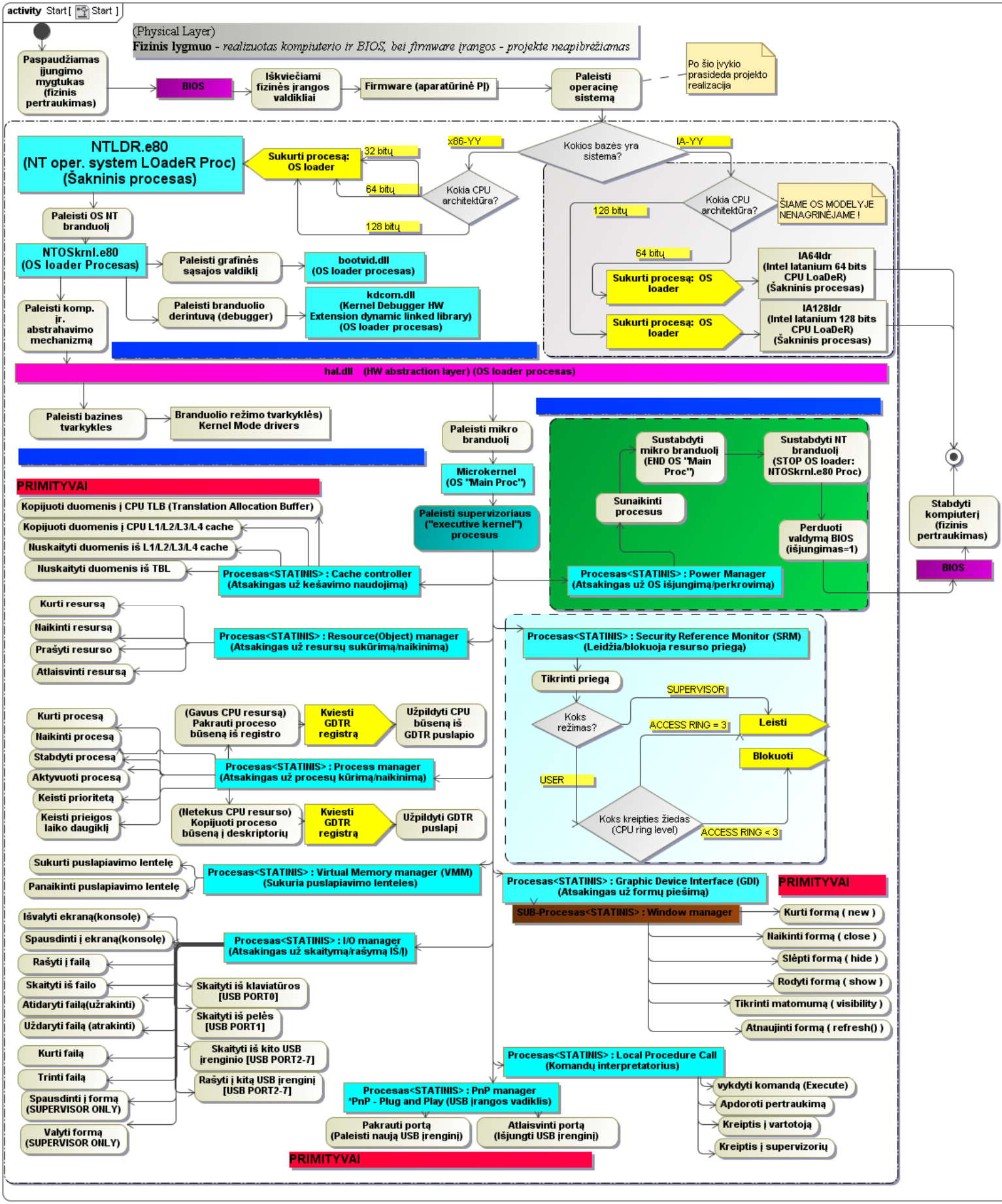


Pavyzdinis, NT sistemos modelis:
(naudojamas Windows NT operacinėje sistemoje(XP ir kt.)



Veiklos diagrama

Kompiuterio darbo rato, nuo paleisties iki išjungimo veiklos ratas (veiklos diagrama) [„Activity diagram“]



OS Branduolio realizacija

1.Branduolio primityvai – procesų, resursų, kešo, buferio, I/O valdiklio, GDI valdiklio.

2.Sisteminiai servisai (angl. „services“) – dar vadinamas „patarnautojais“(angl. *helpers*). Pasikirtis – pagerinti programinės ir techninės įrangos suderinamumą.

3. Procesai – skirstomi į sisteminius ir vartotojo. Procesai – tai atskiros programos(užduotys), šiuo metu veikiančios kompiuteryje.

4. idle – specialus pseudo-procesas, invertuojantis procesoriaus laisvą resursą. „Idle“ veikia nuolat, ir žymi laisvus procesoriaus resursus. Būtent iš šio proceso yra atiminėjami resursai pirmiausia, kol jis jų turi. Ir tik šiam procesui nebeturinti resursų, pradedami blokuoti kiti procesai.

5.Sisteminiai resursai – tai dėl ko varžosi procesai. Resursai gali būti statiniai(sukurti OS įjungimo metu ir gyvuojantys visą laiką) ir dinaminiai(gyvuojantys tam tikrą laiką).

Resursai skirstomi į 4 pagrindes grupes:

1) Integruoti – procesorius(specialus resursas), ram, kietasis diskas
2) Išoriniai – klaviatūra, pelė
3) PnP(plug and play) – USB resursai
4) GDI(graphic device interface) – išvesties(grafiniai ir konsolės) resursai (resursai formose, bei abstraktūs monitoriaus, projektoriaus ir t.t. resursai)

6. Planuotojas*(Task sheduler & scheduling)* - planuotojo paskirtis yra atimti procesorių iš proceso, peržvelgti pasiruošusių procesų sąrašą, išrinkti pasiruošusį procesą, kuris planuotojo manymu yra tinkamiausias, ir perduoti procesorių jam.
Svarbi pastaba: mūsų, ir daugumoje naujų OS planuotojas atlieka tik proceso-kandidato ištrinkėjo vaidmenį, laisvus resursus perduoda procesui jau „idle“ pseudoprocesas. Planuotojas savo veikla išplečia tik tuomet, kad idle informuoja planuotoją apie būtinybę peržiūrėti veikiančius procesus ir iš jų atimti procesorių.

Sistemos primityvai

Primityvai ir specialieji sub-primityvai – nors teoriškai turėtų būti tik šios primityvų grupės: kešo, buferio, resursų, procesų, servisų(mūsų sistemoje nerealizuojami), tačiau praktiškai kuriant operacinę sistemą viską išreikšti per tokius primityvus pasidaro pernelyg abstraktu ir sudėtinga, todėl daugumoje, tame tarpe ir mūsų OS modelyje, į pagalbą yra pasitelkiami specialūs sub-primityvai, kurie kur kas labiau supaprastina sistemos programavimo ir loginio supratimo, darbus.

Procesas-iniciatorius: *„Process Manager“*
Primityvai:

- Kurti procesą
- Naikinti procesą
- Stabdyti procesą
- Aktyvuoti procesą
- ***Keisti prioritetą*** **–** kokią eilę turės procesas „užduočių būgne“, konkuruodamas su kitais procesais
- Keisti prieigos daugiklį - kiek ilgai leisime procesui turėti procesoriui, išskyrus tuos atvejus jeigu procesorius iš proceso būtų atimtas kilus pertraukimui(dėl klaidos ar kito proceso)

Procesas-iniciatorius: *„Service Manager“ (Mūsų OS nerealizuojamas paprastumo dėlei)*
Primityvai:

- Nužudyti servisą *(kill)*
- Stabdyti servisą *(stop)*
- Paleisti servisą *(run)*
- Atidėti servisą *(delay)*

Specialusis procesas-iniciatorius: *„Virtual Memory Manager“*
Specialieji sub-primityvai:

- ***Išskirti puslapiavimo lentelę*** **–** sukurti X-tojo lygio puslapiavimo lentelę su jos vaikais(jei X<7). Šis procesas primityvas visada yra kviečiamas kartu su primityvu „kurti procesą“.
  Pastaba: Moderniose OS yra dar vienas papildomas primityvas „perleisti puslapį“, taip taupant sistemos resursus. Mūsų OS šis dalykas nerealizuojamas.
- ***Panaikinti puslapiavimo lentelę*** **–** sunaikintama X-tojo lygio puslapiavimo lentelė su visais jos vaikais(žemesnio lygio lentelėmis). Šis procesas primityvas visada yra kviečiamas kartu su primityvu „naikinti procesą“.
  Pastaba: Moderniose OS ši funkcija dažnai yra interpretuojama kiek kitaip – tiesiog yra pakeičiamas informacinis bitas puslapių lentelėje, kad šis puslapis yra fiktyvus, tuomet nebereikia atlikti papildomų veiksmų ir taip yra taupomi sistemos resursai.

Procesas iniciatorius: *„Resource Manager“*
Primityvai:

- Kurti resursą
- Naikinti resursą
- Prašyti resurso
- Atlaisvinti resursą

Specialusis procesas-iniciatorius: *„I/O Manager“*
Specialieji sub-primityvai:

- Klausytis (klaviatūros, pelės) („event listener“)
- Skaityti (bylą, kitą USB įrenginį)
- Rašyti (į bylą, į kitą USB įrenginį)
- Kurti (bylą, katalogą)
- Trinti (bylą, katalogą)
- Atidaryti (bylą, katalogą) – užrakinti, paskelbti kvietėją vieninteliu valdovu.
- Uždaryti (bylą, katalogą) – atrakinti.
- Išvesti (į ekraną, į formą(supervisor))
- Išvalyti (ekraną, formą(supervisor))

Specialusis procesas-iniciatorius: *„PnP Manager“*
Specialieji sub-primityvai:

- Atlaisvi portą
- Pakrauti portą

Specialusis procesas-iniciatorius: *„Graphic Device Interface(GDI) Manager“*
Specialieji sub-primityvai:

- Kurti formą *(new)*
- Naikinti formą *(close)*
- Slėpti formą *(hide)*
- Rodyti formą *(show)*
- Tikrinti formos matomumą *(visibility)*
- Perkrauti formą *(refresh)*
- *Perpiešti formą redraw() [mūsų os nerealizuojamas]*

Procesas-iniciatorius: *„Local Procedure call“*
Primityvai:

- Apdoroti pertraukimą
- Interpretuoti komandą
- Perduoti valdymą *(user/supervisor/hypervisor switching)*

Baziniai primityvai*(kaip klasių metodai)*

```
class ProcessManager
{
        [LIST] <...PROCESSES...>
        {
                [OBJ] Process
                [LIST] <...Child Process pointer list...>
                new Process(params ...)
                ~Process()
                void stopProcess()
                void runProcess()
                void changePriority(int newPriority)
                void changeTimeMultipier(double newMultiplier)
        }
        int createProcess(); // RET: processID
        void removeProcess(int processID);
        void changePriority(int processID, int newPriority);
        void changeTimeMultipier(int processID,double newMultiplier);

}

class ResourceManager
{
        [LIST] <...RESOURCES...>
        {
                [OBJ] Resource
                [LIST] <...Resource waiting processes...>
                new Resource(params ...);
                ~ Resource ();
                void getResource();
                void freeResource ();
        }
        int createResource(params ...); // RET: resourceID
        void removeResource (int resourceID);
        void getResource(string resourceType);
        void freeResource (int resourceID);
        abstract bool getResourceOrWait(); // RET: Wait: false, Else: True

}
```

Procesų valdymo notacijos:

fork() – funkcija sukuria einamojo proceso kopiją, kuri nuo motininio proceso skiriasi tik identifikatoriumi. Skirtingai nuo kitų C kalbai būdingų funkcijų, ši grąžina du rezultatus: vieną – į motininį procesą (sukurto proceso identifikatorių), kitą – į dukterinį procesą (nulį).

exec() – negrąžina jokio rezultato, bet einamąjį procesą pakeičia į kitą, naujai sukurtą iš vykdomojo failo, kuris nurodytas šiai funkcijai.

wait() – laukia, kol ***fork()*** sukurtas procesas pasibaigs, naudojant ***exit()*** funkciją, bei grąžina rezultatą, perduotą ***exit()*** funkcijai.

exit() – baigia procesą ir grąžina rezultatą motininiam procesui.

Baziniai procesai

Terminai:
***dll** – dinaminė biblioteka (dynamic linked library)
***.e80** – emu128 multiprogramines operacinės sistemos plėtinys *(kitose OS jis gali būtų pvz. *.exe ir pan.)*.
***NT** – nereiškia santrumpos NeTwork!. Ši santrumpa žymi frazę *„N-Ten“*, kuri yra aiškinama kaip *„New Technology“* trumpinys.
*.**sys** – sisteminis procesas (SYStem process)

Sisteminiai procesai:

1. **NTLDR.E80** *(NT operating system LoaDeR process)* – šakninis procesas. Terminologiškai vadinamas **„StartStop“** procesu. Šis procesas aktyvuojamas DOS režime.
2. **ntoskernel.e80** *(NT OS kerner loader)* – procesas užkraunantis NT failų sistemą, suskirstantis atmintį į puslapius ir paleidžiantis 3 bazinius procesus.
   2.1. **bootvid.dll** – procesas, atsakingas už grafinės sąsajos paleidimą.
   2.2. **kdcom.dll** *(Kernel Debugger HW extension DLL)* – branduolio derinimo-valdymo procesas-biblioteka. Aptarnaujantis derinimo režimą, derinimo registrus ir STACK CALLBACK mechanizmą derinimo režime).
   2.3. **hal.dll** *(Hardware Abstraction layer)* – valdo įvesties/išvesties sąsajas*(interfaces)*, pertraukimų valdiklius*(interrupt controllers)* ir kelių branduolių procesorius. Šis procesas sukuria procesą „*Microkernel.e80*“.

4.**mkernel.e80** *(Operating system „Main Proc“ for 1 core)* – terminologijoje dažnai apibūdinamas kaip **„Main Proc“**

3. **svchost.e80** (switch host) – resursų paskirstytojas, dalijantis sisteminius resursus(atmintį, procesorių, magistralės priegą(„system bus“)).
4. **services.e80** (Service Control Manager (SCM)) – MŪSŲ SISTEMOJE ESAME VARIANTE NENAUDOJAMAS. Tačiau šis procesas yra labai dažnas daugelyje NT OS – tai „Remote procedure call“ serveris. Jis atsakingas už naujų OS servisų paleidimą, išjungimą, ištrynimą, sukūrimą. Daugumos servisų esmė – galimybė dirbti tinkle, ir valdyti sistemą nuotoliniu(„remote“) būdu. Taip pat servisai atsakingi už geresnį „shared“ įrangos palaikymą – kaip skaneriai, spausdintuvai, faksai ir kt.
5. **core.sys** – terminologijoje vadinamas „**planuotoju**“, kartu dar ir įtraukiant „**jobGovernor**“ aprašą. Mūsų architektūroje į šį procesą yra įtrauktas procesas ***Microkernel.e80***, t.y. šis procesas kuria virtualias mašinas(procesus, o vartotojo režime - programas), skirsto joms prioritetus, valdo eigą tarp sisteminės magistralės ir procesų, siunčia blokavimo/atblokavimo signalus, priskiria resursus. Į šį procesą kreipiasi virtualios mašinos su savo metodu „*getResourceOrWait()*“, o šis procesas savo ruožtu kreipiasi į resourceManager, kviečia procesą ***io.sys***.
   Taip pat šis procesas mūsų sistemoje apima ir GDI/Windows Manager funkcija, t.y. formų kūrimą, trynimą, naujinimą, slėpimą, rodymą, matomumo tikrinimą. Šis procesas taip pat rūpinasis ir saugumu, t.y. atlieka SRM*(Security Reference Monitor)* funkcijas, t.y. daro sutirkinimus tarp kreipinio ir atsako apsaugos lygių(CPU rings). Per šį procesą kreipiamasi į MemoryManager, BusManager, RAM, VirtualMemoryPaging objektus. Per šį procesą taip pat yra bendraujama su komandų interpretatoriumi. Šis procesas taip pat inicijuoja kešavimą ir kešo atkūrimą į CPU registrus.
6. **io**.**sys** – procesas atsakingas bet kokį įvedimą/išvedimą į/iš USB portus, konsolę, ir formas(SUPERVIZORIAUS režime).

Vartotojo procesai:

Iki 32 vartotojo procesų (taktinėje magistralėje žymimų #1- #32 programomis).

#0 – specialus vartotojo procesas, žymimas „System Idle process“. Žymintis laisvus resursus. Vykstantis nuolatos. Išjungus blokuojasi visi laisvi resursai.

Baziniai operacinės sistemos mechanizmai

Kompiliatorius *(objektas)*:

Bazinis objektas, komandų interpretavimui. Tikrina ar komanda egzistuoja, tikrina ar esame reikiamame bloke, bendrauja su JumpManager objektu, gražina kompiliavimo ir logines klaidas.

Pertraukimų mechanizmas *(objektas)* (***Interrupt controller***, mūsų sistemoje pavadintas ***ExceptionHandler***)**:**

Įrenginys(mūsų OS atveju – objektas-klasė), apdorojantis sisteminius petraukimus(unhandled exceptions).

Pertraukimai gali kilti, kai:

1. Laukiama įvesties iš vartotojo ekrano – ‚užšaldoma' einama programa, kitos programos vykdomos toliau.

2. Nebuvo gautas procesoriaus resursas – procesui nurodoma laukti.

3. Įvyksta programinis pertraukimas - Buvo įvykdyta „return", „callback", „halt", „lock", „intXXh" – komanda.

4. Įvyksta nenumatytas programinis pertraukimas – sintaksės klaida.

5. Įvyksta nenumatytas programinis pertraukimas – logikos klaida. (pvz. bandymas pasiekti neegzistuojančią atminties vietą).

6. Randamas programos pabaigos ženklas.

7. Įvyksta sisteminė klaida (*„hard interrupt"*) – užšaldoma ir sustabdoma visa sistema ir visos joje veikusios programos. Sistema terminuojama, arba yra sustabdoma dirbant derinimo režimu.

8. Gaunamas sistemos išjungimo signalas.

9. Įvyksta perpildymas (overflow) – registro arba steko perpildymas.

10. Įvyksta „OutOfMemoryBounds" klaida – neužteko puslapių lentelės dydžio, arba nepavyko sukurti atitinkamos laisvos erdvės fizinėje RAM atmintyje.

11. Įvyksta blokavimai kažko(resurso ar proceso) laukiant – išsekus taimeriui.

Sinchronizavimas ir multioperaciškumas

Sinchronizacija ir multioperaciškumas sistemoje įgyvendinamas naudojantis Mutex'u bei įvedus kritinės sekcijos sąvoką.

Mutex'as turi du primityvus – **signal(s)** ir **wait(s)**. 2 procesai vienu metu negali būti ***wait(s)*** arba ***signal(s)*** viduje. Mutex'as riboja, kad vienu metu tik vienas procesas gali patekti į kritinę sekciją. Šitas laukimas neturi tęstis neapibrėžtą laiką.

Kritinė sekcija(KS) (angl. CS) - tai programos dalis, kurioje atliekami veiksmai su bendrai naudojamais duomenimis. Kad programa duotų teisingus rezultatus, reikia užtikrinti, kad kritinėje sekcijoje tuo pačiu metu būtų ***tik vienas procesas***.

Kritinės sekcijos problema:

- Kai procesas vykdo kodą, kuris susijęs su bendrais duomenimis, sakome, kad procesas *yra kritinėje sekcijoje* (CS) (tiems duomenims)
- Kritinės sekcijos vykdymas turi būti atliekamas *mutex* (tarpusavio išskirtinumo) sąlygomis: bet kuriuo metu tik vienas procesas gali būti kritinėje sekcijoje (nesvarbu kiek CPU)
- Kiekvienas procesas turi prašyti leidimo įeiti į kritinę sekciją.

Geras būdas kritinei sekcijai tvarkyti - **semaforų** naudojimas.

Semaforas S yra skaitinis, sveikąsias teigiamas reikšmes įgyjantis, kintamasis, kur be inicializavimo, galima jį veikti per dvi atomines ir mutex'ines operacijas - wait(s) ir signal(s).
Tam, kad išvengti busy-waiting. kai procesas turi laukti, tai semaforas gali patalpinti jį į blokuotą procesų eilę.

Semaforą galima aprašyti taip:

```
type semaphore = record
           count : integer;
           queue : list of process
           end;
var S : semaphore
```

signal(s) paleidžia ką nors iš eilės, o ***wait(s)*** padeda į eilę.

Semaforų apžvalga:

- Kai S.count ≥ 0, tai tas kiekis procesų gali įvykdyti wait(s) be blokavimo.
- Kai S.count < 0, tai | S.count | rodo eilės ilgį.
- Atomiškumas: wait(s) ir signal(s) atominės ir mutex'inės : 2 procesai vienu metu negali būti wait(s) ar signal(s) viduje.

Semaforo S reikšmes nustato du primityvai - **P** ir **V:**

V(S): nedalomu veiksmu **S** reikšmė padidinama 1;

P(S): jei galima, nedalomu veiksmu **S** reikšmė sumažinama 1; jei **S=0**, tai **S** sumažinti negalima, todėl procesas laukia, kol sumažinimas bus galimas.

Gija - tai sąlyginai savarankiškas programos fragmentas.
Kiekvienas procesas gali būti suskaidytas į atskiras gijas.
Tokiu atveju uždavinys gali būti išspręstas greičiau, panaudojant pseudolygiagretų (vienprocesorinėje sistemoje) arba lygiagretų (multiprocesorinėje sistemoje) jos atskirų dalių vykdymą.
Tradicinėse OS **gijos** sąvoka atitinka **proceso** sąvoką. Tačiau skirtingos gijos vieno proceso ribose mažiau nepriklausomos lyginant su skirtingais procesais:
- visos tokios gijos turi tą pačią adresų erdvę,
- nėra reikalo apsaugoti vieną giją nuo kitos,
- tokios gijos sprendžia tą pačią vartotojo užduotį.

Failų sistemos - ZFS *(Zettabyte File System)* - principai

Terminai:

VD, vdev - virtualūs įrenginiai (Virtual Devices)
POSIX - bendras vardas IEEE specifikuotų susijusių standartų rinkiniui, skirtų apibrėžti „aplikacijų programavimo sąsajos“(API) taikomųjų programų suderinamumui tarp įvairių OS variantų užtikrinti.

0. Įvadas

ZFS – tai šiuo metu nemokama, Open Solaris organizacijos kuriama ir tobulinama failų sistema.
ZFS – tai nauja failų sistemos technologija, kurios dėka galima pasiekti milžinišką talpą(128-bitų), pagerintą duomenų integruotumą, savaimio-optimizavimosi našumą bei realaus laiko nuotolinį kopijavimą.

ZFS atsiskiria nuo standartinio failų sistemos modelio eliminuodamas skirsnio koncepcijas. Vietoje to, ZFS failų sistema dalinasi bendrą saugyklos „baseiną“ susidedantį iš įrašomos disko terpės.

ZFS programinė įranga susideda iš 7 skirtingų detalių:

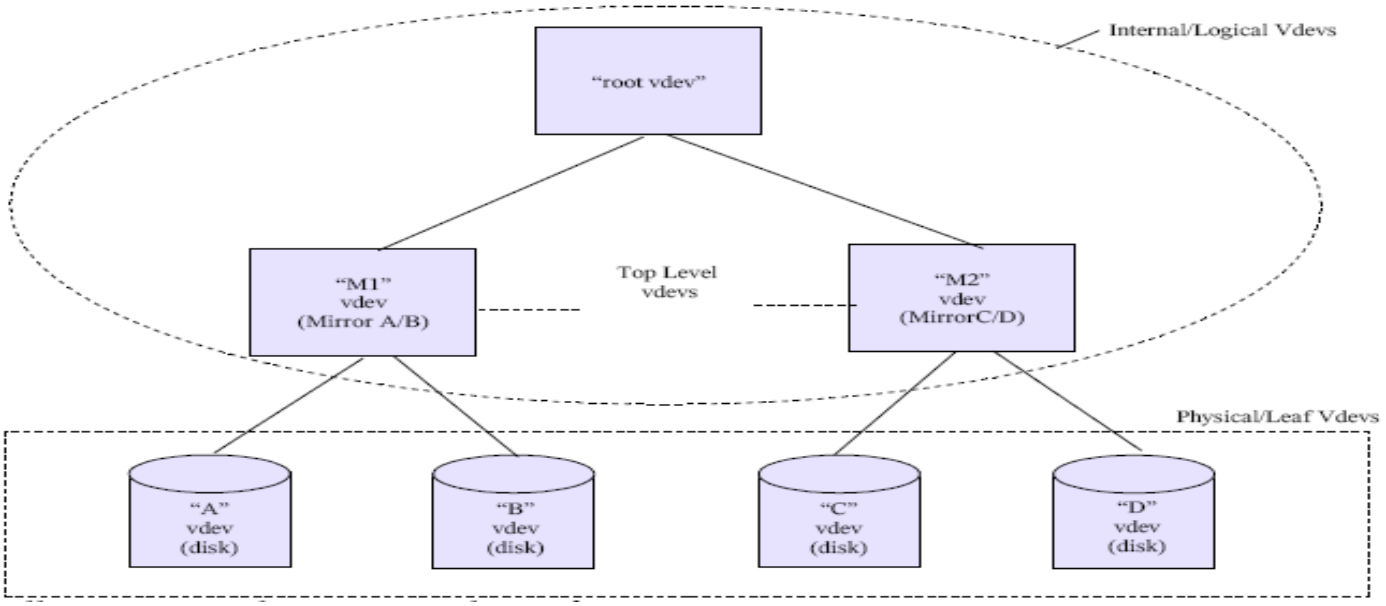
1. ***SPA*** *(Storage Pool Allocator)* – saugyklos baseino išskyrėjas
2. ***DSL*** *(Dataset and Snapshot Layer)* – duomenų sąrašo ir atvaizdo sluoksnio
3. ***DMU*** *(Data Management Layer)* – duomenų valdymo sluoksnio
4. ***ZAP*** *(ZFS Attribute Processor)* – ZFS atributų apdorojimo procesoriaus
5. ***ZPL*** *(ZFS Posix Layer)* – ZFS Posix sluoksnio
6. ***ZIL*** *(ZFS Intent Log)* – ZFS ketinimų žurnalo
7. ***ZVOL*** *(ZFS Volume)* – ZFS skirsnio

ZFS naudoja „kopijuoti-rašant“ (COW – Copy on Write) transakcijos modelį. Šis modelis užtikrina, kad duomenys niekada nėra perrašomi ir visi disko atnaujinimai vyksta atominių operacijų pagrindu.

1. Virtualūs įrenginiai

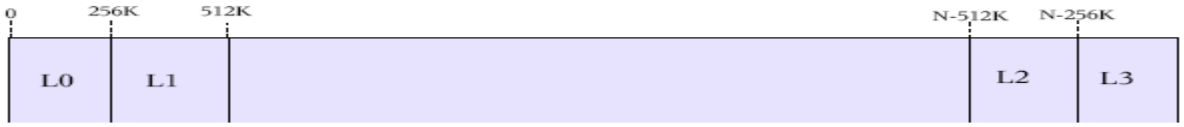
ZFS duomenų saugyklos baseinai yra sudaryti iš virtualių įrenginių(VD) kolekcijos.
VD yra dviejų tipų:
FVĮ – fiziniai virtualūs įrenginiai*(kartais vadinami „lapų VD‘sais“)*. Tai įrašomos multimedijos įrenginys (pvz. kietasis diskas).
LVĮ – loginiai virtualūs įrenginiai*(kartais vadinami „vidiniais VD‘sais“)*. Tai abstraktus FLĮ grupavimas.

VD‘sai yra išdėstyti medžio(tree-view) principu. Kiekvienas „baseinas“ turi „šakninį“ (root) VD‘są(fizinį arba loginį) ir vadinama aukščiausio lygmens VD‘su(top-level vdevs).



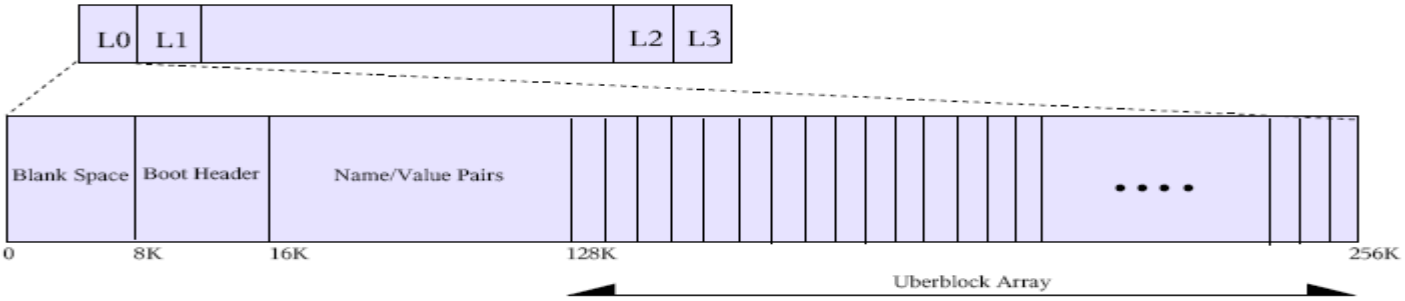
2. Virtualių įrenginių antraštės (Vdev Labels)

Kiekvienas FVĮ su duomenų saugyklos baseinu, turi 256KB dydžio struktūra pavadinta „vdev antrašte“. Pvz. „C“ disko antrašė paveikslėlyje turės informaciją apie „C“, „D“ bei „M2“.
Šių antraščių paskirtis – suteikiama prieiga pasiekti „baseino“ turinį, bei patvirtinti „baseino“ integruotumą ir prieinamumą.
Po 4 kiekvienos antraštės kopijas yra saugoma kiekviename FVĮ-io ZFS duomenų saugyklos „baseine“.



VĮ antraštės išsidėstymas N dydžio bloke(įrenginyje).

Kiekviena iš antraščių susideda iš blokų:



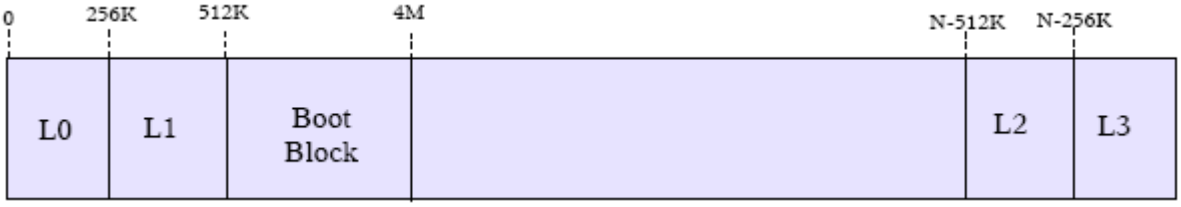
1) 8KB Tuščios vietos (Blank Space) – ZFS supranta abu VTOC *(Volume Table of Contents)* bei EFI metodus kaip korektiškus metodus apibrėžiančius disko struktūrą.
2) 8KB įkrovos antraštinės informacijos (Boot Header)
3) 112KB pavadinimas-reikšmė porų – apibudina šakninį, kaimyninį ir vaikinius virtualius įrenginius.

3.Uber-blokas („Uberblock“)

Tai antraštės dalis susidedanti iš informacijos, būtinos norint pasiekti „baseino“ turinį. Tik vien uber-blokas blokas baseine yra aktyvus vienu metu.

4. Įkrovos blokas („boot block“)

3.5MB dydžio įkrauties blokas. Struktūra:



VĮ antrašės išsidėstymas įskaitant įkrovos(boot) bloko rezervuotą vietą.

Paimkime pavyzdžiui, VĮ disko „vdev A“, „nvlist“ įrašą ir panagrinėkime detaliau:

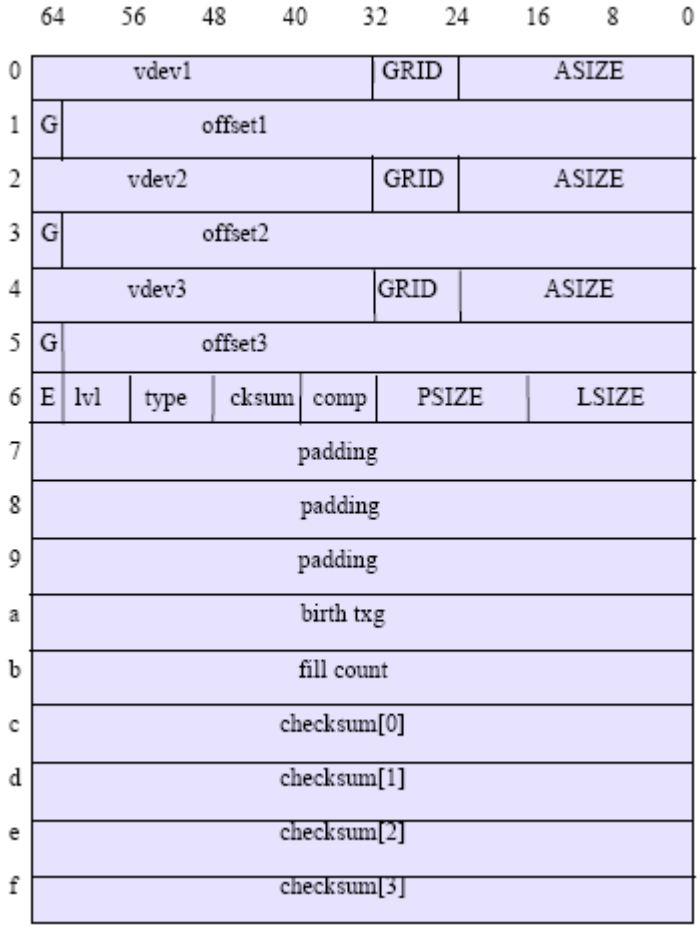
1. Laukelis **„type“** *(TYPE)*:
   apibūdina virtualaus įrenginio rūšį:

Tipas	**Reikšmė** *(DATA_TYPE_STRING)*
disk	Lapo VĮ: blokų saugykla („block storage“)
file	Lapo VĮ: failų saugykla („file storage“)
mirror	Vidinis VĮ: veidrodis („mirror“)
raidz	Vidinis VĮ: masyvas („raidz“)
replacing	Veidrodinis VĮ: naudojamas ZFS keičiant vieną diską kitu
root	Vidinis VĮ: VĮ medžio šaknis

2. Laukelis **„id“** *(ID)*: *{DT: DATA_TYPE_UINT64}.*
   Šio VĮ indeksas jo tėvinio VĮ vaikų sąraše.
3. Laukelis „**guid**“ *(GUID - Global Unique Identifier)*: *{DT: DATA_TYPE_UINT64}*
   Globalus šio VĮ medžio elemento identifikatorius..
4. Laukelis **„path“**: *{DT: DATA_TYPE_STRING }*
   Kelias iki VĮ. Naudojama tik „lapiniams“ VĮ.
5. Laukelis **„devid“** *(DEVID – Device ID):* *{DT: DATA_TYPE_UINT64}*
   Įrenginio ID VĮ medyje. Naudojamas tik „disk“ tipo VĮ medžio elementams.
6. Laukelis **„metaslab_array“**: *{DT: DATA_TYPE_UINT64}*
   Numeris objekto, susidedančio objektų numerių masyvo.
7. Laukelis **„metaslab_shift“**: *{DT: DATA_TYPE_UINT64}*
   Logaritmas pagrindu 2 „metaslab“ dydžiui.
8. Laukelis **„ashift“**: *{DT: DATA_TYPE_UINT64}*
   Globalus šio VĮ medžio elemento identifikatorius.
9. Laukelis **„asize“** *{DT: DATA_TYPE_UINT64}*
   Kiek vietos gali būti išskirta iš šio lygio VĮ.
10. Laukelis **„children“**: *{DT: DATA_TYPE_NVLIST_ARRAY }*
    „nvlist“ elementų masyvas, sudarytas iš kviekvieno vaikinio elemento.

2. Blokų rodyklės(„pointers“) bei netiesioginiai blokai(„indirect“)

Duomenys tarp disko ir pagr. atminties yra perkeliama vienetais, pavadintais „blokais“. Bloko rodyklė „**blkptr_t**“ – tai 128 baitų ZFS struktūra naudojama fiziškai surasti, patvirtinti ir aprašyti duomenų blokus diske.
blkptr_t struktūrą matome žemiau:



2.1: DVA – Duomenų virtualus adresas (Data Virtual Address)

DVA – tai vardas, suteiktas kombinuojant VĮ ir poslinkio(offset) porcijas bloko rodyklėje. Pvz. vdev1 ir offset1 sukombinavus būtų DVA(dva1). ZFS suteikia galimybę saugoti iki 3 duomenų kopijų, į kurias rodo bloko rodyklės, kurių kiekviena unikali DVA(dva1, dva2 arba dva3). Duomenys išsaugoti kiekvienoje iš šių kopijų yra identiški. DVA‘ų skaičius naudojamas bloko rodykle yra grynai ‚politinis‘ sprendimas: vienguba plati bloko rodyklė(1 DVA‘ų), dviguba plati bloko rodyklė(2 DVA‘ų) ir triguba plati bloko rodyklė(3 DVA‘ų).
Kiekviena VĮ dalis kiekvieno DVA‘o yra 32 bitų integer tipo skaičius, kuris unifikuoja VĮ ID esančio šiame bloke. Poslinkio dalis kiekviename DVA yra 63 bitų ilgio integer tipo skaičius(pradedant iškart po VĮ antraščių(L0 ir L1) ir įkrovos bloko). Kartu vdev+offset unikaliai identifikuoja duomenų bloko adresą.

3.DMU - Duomenų valdymo vienetas (Data Management Unit)

DMU sunaikina blokus ir sugrupuoja juos į loginius vienetus vadinamus objektais. Vėliau DMU dėka jie gali būti sugrupuoti į objektų sąrašus.

4. DSL – Duomenų sąrašo ir atvaizdo sluoksnis (Dataset and Snapshot Layer)

DSL pateikia mechanizmą aprašyti ir valdyti tarpusavio sąryšius ir savybes tarp objektų sąrašų.
Objektų sąrašo apžvalga:
ZFS leidžia sukurti 4 rūšių objektų sąrašus: failų sistemas, klonus, atvaizdus bei skirsnius.
ZFS failų sistema – saugo ir organizuoja lengvai prienama, POSIX atitinkančia, forma.
ZFS klonas – indentiškas failų sistemai.
ZFS atvaizdas – tik-skaitymui(read-only) tipo failų sistemos versija, klonas arba skirsnis tam tikru laiko momentu.

ZFS skirsnis – loginis skirsnis ZFS eksportuotas kaip bloko įrenginys.

5. ZAP – ZFS atributų procesorius (ZFS atribute processor)

ZAP – tai modulis, esantis DMU viršūnėje ir operuojantis objektai, pavadintais ZAP objektais. ZAP objektas – tai DMU objektas skirtas saugoti atributus vardas-reikšmė porų formatu. Vardo dalis yra iki 256 baitų(įskaitant NULL pabaigos simbolį).
ZAP objektai skirti saugoti duomenų sąrašų savybes, naviguoti failų sistemų objektais, saugoti baseinų savybes ir kt.

ZAP objektų tipai:

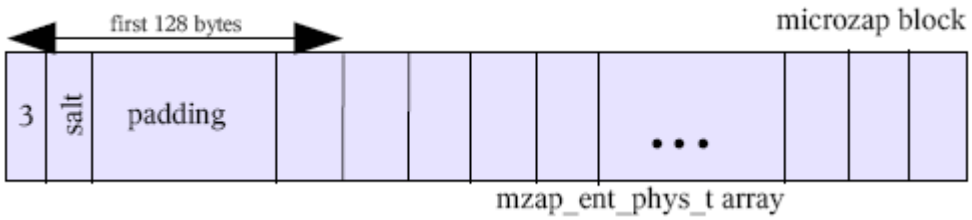
- DMU_OT_OBJECT_DIRECTORY
- DMU_OT_DSL_DIR_CHILD_MAP
- DMU_OT_DSL_DS_SNAP_MAP
- DMU_OT_DSL_PROPS
- DMU_OT_DIRECTORY_CONTENTS
- DMU_OT_MASTER_NODE
- DMU_OT_DELETE_QUEUE
- DMU_OT_ZVOL_PROP

ZAP objektai yra 2 tipu – microzap ir fatzap tipo objektai. Microzap objektai yra „lengvos“ fatzap objektų versijos, ir pateikia paprasta mechanizmą nedidelio kiekio atributų įrašų paieškai.
Fatzap labiau tinkamas ZAP objektams susidedantiems iš didelio kiekio atributų.

ZFS sprendžia kurį rinktis(microzap ar fatzap) remdamasis:
Visos vardas-reikšmė poros telpa viename bloke(128KB, t.y. 2047 microzap įrašai).
Visų atributų reikšmė yra uint64_t. tipo.
Vardas yra lygus arba trumpesnis nei 50 simbolių(įskaitant NULL pabaigos simbolį).
Jeigu nors viena sąlyga netenkinama, naudojamas fatzap.
Pirmas 64 bitų žodis kiekviename ZAP objekto bloke skirta identifikuoti ZAP objekto turinio tipą.

5.1. The Micro Zap

Microzap realizuoja paprastą mechaniza nedideliam kiekiui atributų saugoti. Microzap objektas susideda iš vieno bloko, kuriame yra ***mzap_ent_phys_t*** strukturos elementų masyvas. Kiekvienas atributas išsaugotas microzap objektai yra reprezentuojamas kaip vienos iš šių struktūrų.
Pirmi 128 bitai sudaro mzap_phys_t įrašą, kuriame įtraukta 64 bitų ZBT_MICRO reikšmė, definuojanti kad šis blokas yra skirtas saugoti microzap tipo įrašus. 42 baitai palikti laisvi, ir paskutiniai 64 baitais susideda iš microzap ***mzap_ent_phys_t*** struktūros tipo įrašo:



Microzap bloko išdėstymas (layout)

5. Rodyklių lentelė (Pointer Table)

Raktų lentelė, naudojanti grandinės metodą valdyti kolizijas. Kiekvienas įrašas(„hash“) susideda iš 64-bitų sveikojo „integer“ tipo skaičiaus, kuris apibūdina 0-lygio bloko ID.

6. ZPL – ZFS POSIX sluoksnis (ZFS POSIX Layer)

ZPL paverčia DMU objektus atrodyti kaip POSIX failų sistemoje. ZPL reprezentuoja failų sistemą kaip ***DMU_OST_ZFS*** tipo objektų seką. Visi atvaizdai, klonai ir failų sistemos yra realizuotos kaip šio tipo objektų sekos.

6.1. ZFS prieigos teisių sąrašai (ACL - Access Control Lists)

ACL tarnauja kaip mechanizmas, leidžiantis apriboti arba leisti vartotojo prieigą prei ZFS objektų. ACL sąrašai yra realizuoti ZFS sistemoje kaip lentelė iš ACE įrašų(Access
Control Entries – prieigos teisių įrašai):

Atributas	***Reikšmė***
ACE_READ_DATA	0x00000001
ACE_LIST_DIRECTORY	0x00000001
ACE_WRITE_DATA	0x00000002
ACE_ADD_FILE	0x00000002
ACE_APPEND_DATA	0x00000004
ACE_ADD_SUBDIRECTORY	0x00000004
ACE_READ_NAMED_ATTRS	0x00000008
ACE_WRITE_NAMED_ATTRS	0x00000010
ACE_EXECUTE	0x00000020
ACE_DELETE_CHILD	0x00000040
ACE_READ_ATTRIBUTES	0x00000080
ACE_WRITE_ATTRIBUTES	0x00000100
ACE_DELETE	0x00010000
ACE_READ_ACL	0x00020000
ACE_WRITE_ACL	0x00040000
ACE_WRITE_OWNER	0x00080000
ACE_SYNCHRONIZE	0x00100000

7 ZIL – ZFS ketinimų žurnalas (ZFS Intent Log)

ZIL žurnale saugomi sistemos kvietimų transakcijų įrašai, kurie keičia failų sistemą atmintyje. Log‘ai yra saugomi atmintyje kol DMU transakcijų grupė(txg) nusprendžia juos išsaugoti kaip įrašus diske(taip pat baseine) ***fsync*** ar ***O_DSYNC*** metu arba atmesti ir ištrinti. Klaidos atveju šie įrašai yra atkuriami vietoje esamų.
Yra tik vienas ZIL žurnalas, susidedantis iš 3 dalių:

- ZIL antraštės
- ZIL blokų
- ZIL įrašų

ZIL blokai susideda iš ZIL įrašų. Išskiriami pagal pareikalavimą ir yra kintamamo dydžio pagal poreikį. Dydžio laukelis yra dalis ***blkptr_t*** rodyklės, kuri rodo į žurnalo bloką. Kiekvienas blokas yra užpildytas įrašais ir turi ***zil_trailer_t*** atžymą kiekvieno bloko gale.

8 dalis – ZVOL (ZFS Volume)

ZFS skirsniai pateikia mechanizmą kurti loginius skirsnius. ZFS skirsniai yra eksportuojami kaip blokiniai įrenginiai ir gali būti naudojamas kaip ir bet koks kitas blokinis įrenginys. ZVOL skirsniai yra aprašyti *DMU_OST_ZVOL* objektų sąraše. ZVOL objektas yra labai paprasto formato, susidedančio iš 2 objektų: savybių*(DMU_OT_ZVOL_PROP)* ir duomenų*(DMU_OT_ZVOL)* objektų. Abu turi statiškai priskirtus objektų ID.

Kodo formatas

```
extrn papildinio_pavadinimas : proc

strukturos_pavadinimas struct
...
strukturos_pavadinimas ends

.const
konstantos_pavadinimas equ 21h
...

.data
...

.code
programos_pavadinimas:
...
PROC proceso_pavadinimas:
...
ENDPROC proceso_pavadinimas
...
end programos_pavadinimas
```

Programos pavyzdys(x86-64+):

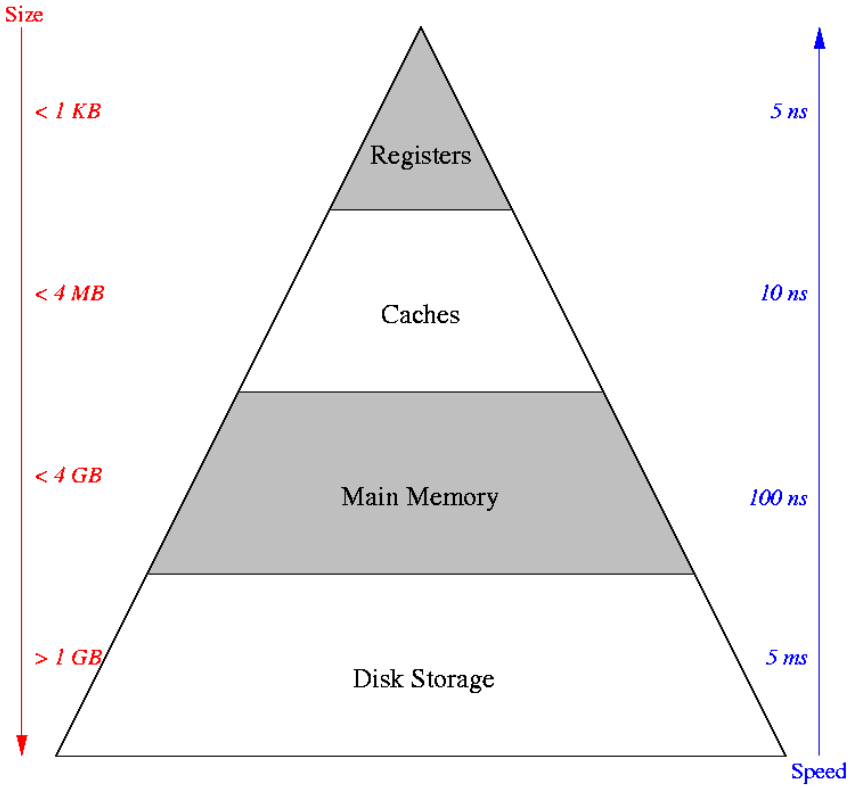
```
extrn MessageBoxA : proc
extrn ExitProcess : proc

POINT struct
  x                 dd      ?
  y                 dd      ?
POINT ends

MSG struct
  hwnd              dq      ?
  message           dd      ?
  padding1          dd      ?       ; padding
  wParam            dq      ?
  lParam            dq      ?
  time              dd      ?
  pt                POINT   <>
  padding2          dd      ?       ; padding
MSG ends

.const
NULL equ 0
DO equ 21h

.data
body db 'Sveiki!', 0
capt db 'Tai pirma x64 programa', 0    ; komentaras

.data?
hInstance qword ?
hWnd qword ?
wmsg MSG <>

.code
Main proc
sub rsp, 28h
xor r9d, r9d        ; uType = 0
lea r8, capt        ; lpCaption
lea rdx, body       ; lpText
xor rcx, rcx        ; hWnd = NULL
call MessageBoxA
xor ecx, ecx        ; exit code = 0
call ExitProcess
Main endp

end
```

Procesorių, realizuojančių x86 instrukcijų rinkinį(instruction set), sąrašas*(desktop PC)*:

Kiekviena x86 procesorių karta*(Generation)* remiasi naujesniu instrukcijų rinkiniu.

Karta *(Gen.)*	**Metai**	**Procesoriai (CPU)**	**tiesiog. / fiz. adresavimo erdvė** *(linear / physical address space)*	**Palaikomi Instrukcijų rinkiniai / Savybės**
1	1978	Intel 8086 / 8088	**16-bitų** *fizinė* / **20-bitų** *segment.*	**I.R.:** Standartinis x86 16bitų instrukcijų rinkinys
2	1982	Intel 80186 / 80188		
		Intel 80286	**16-bitų** *fizinė* / **30-bitų** *virtuali* / **24-bitų** *segment.*	**I.R.:** x86 16 bitų **Savybės:** MMU
3 *IA-32*	1985	Intel386, AMD Am386	**32-bitų** *fizinė* / **46-bitų** *virtuali*	**I.R.:** 32-bitų instrukcijos **Savybės:** MMU su puslapiavimo mechanizmu
4	1989	Intel486, AMD Am486		**Savybės:** integruotas FPU, L1
5	1993	Pentium I / MMX		**I.R.:** MMX **Savybės:** 64 bitų magistralė
6	1995	AMD K5	**32-bitų** *fizinė*	**I.R.:** MMX
		Pentium Pro	**32-bitų** *fizinė* / **36-bitų** *PAE*	**I.R.:** MMX **Savybės:** PAE, integruota L2
	1997	AMD K6-1 / K6-2 / K6-3, Pentium II/III		**I.R.:** MMX, 3DNow, SSE **Savybės:** L3
7	1999	Athlon, Athlon XP		**I.R.:** MMX, 3DNow, SSE
	2000	Pentium 4		**I.R.:** MMX, SSE, SSE2 **Savybės:** aukšto takt. dažnio *(iki 3.4Ghz)*, HT
8 *x86-64*	2003	Athlon 64	**64-bitų** *fizinė.*	**I.R.:** x86-64 instrukcijų rinkinys, HT
	2004	Pentium Prescott,		**I.R.:** SSE3 **Savybės:** itin aukšto takt.dažnio *(3-3.8 Ghz)*,
9	2006	Intel Core 2		žemo taktinio dažnio *(1,8-3 Ghz)*, SSSE3
10	2007	AMD Phenom	**64-bitų** *virtuali* / **48-bitų** *fizinė*	128 bit FPUs, SSE4a, L3 kešas
	2008	Intel Core i7	**64-bitų** *fizinė.*	**I.R.:** SSE4.1,SSE4.2, AES, CLMUL *(Wesmere)* 128 bit FPUs, SS4.1, HT 3/QuickPath, L3 kešas
11 *x86-128*	2011	AMD Bulldozer	**128-bitų** *fizinė*	**I.R.:** SSE5, AES
		Intel Sandy Bridge,		**I.R.:** AVX

Našumo hierarchija



Šaltiniai

Internete:

a. HDD išleidimų istorija:
http://www.alts.net/ns1625/winchest.html
b. NX BIT:
http://en.wikipedia.org/wiki/NX_bit
c. Žiedų sistema:
http://en.wikipedia.org/wiki/Ring_%28computer_security
d. NT sistemos architetūrą:
http://en.wikipedia.org/wiki/Architecture_of_Windows_NT
e. NT sistemos paleisties procesas:
http://en.wikipedia.org/wiki/Windows_NT_startup_process
f. Apie puslapiavimo lentels:
http://www.pagetable.com/
g. Puslapiavimo mechanizmas:
http://www.sandpile.org/
h. Boot sektorius kietąjame diske:
http://en.wikipedia.org/wiki/Master_boot_record
i. Virtuali mašina:
http://en.wikipedia.org/wiki/Virtual_machine
j. Vektorinis procesorius:
http://en.wikipedia.org/wiki/Vector_processor
k. x86 instrukcijos:
http://en.wikipedia.org/wiki/X86_instruction_listings
l. AMD64 (EM64T) architecture:
http://www.viva64.com/content/articles/64-bit-development/?f=amd64_em64t.html&lang=en&content=64-bit-development
m. ZFS:
http://en.wikipedia.org/wiki/ZFS
n. NTFS:
http://en.wikipedia.org/wiki/NTFS
o. Comparison of file systems:
http://en.wikipedia.org/wiki/Comparison_of_file_systems
p. Branduolių tipai:
http://en.wikipedia.org/wiki/Monolithic_kernel
http://en.wikipedia.org/wiki/Hybrid_kernel
http://en.wikipedia.org/wiki/Microkernel
q. Žiedų sistema:
http://en.wikipedia.org/wiki/Ring_(computer_security)
r. x86 virtualizacija:
http://en.wikipedia.org/wiki/X86_virtualization
s. IPC:
http://en.wikipedia.org/wiki/Inter-process_communication
t. Protected mode:
http://en.wikipedia.org/wiki/Protected_mode
u. POSIX:
http://lt.wikipedia.org/wiki/POSIX
v. Windows achitechture:
http://en.wikibooks.org/wiki/Windows_Programming

PDF, PPT, DOC(intel.com, amd.com ir kiti):

a. *(Intel)* Intel Core 2 new_architecture_06.pdf
b. *(Intel)* Core i7 x64 instruction M-Z.pdf
c. *(Intel)* AVX_Intel Advanced Vector Extensions Programming Reference-006.pdf
d. *(Intel)* Intel® 64 and IA-32 ASD Manual Vol1 - Basic Achitectures.pdf
e. *(Intel)* Intel 64 Sofware Development Manual Vol1 - A-M.pdf
f. *(Intel)* Intel 64 Sofware Development Manual Vol2 - N-Z.pdf
g. *(Intel)* Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3A - System Programming Guide, Part 1.pdf
h. Address Translation with Paging.pdf
i. Memory_Management_From_Absolute_Addresses_to_Demand_Paging.pdf
j. *(AMD)* AMD 128-256 FMA4 and XOP instructions.pdf
k. *(AMD)* AMDx86-64_overview.pdf
l. *(AMD)* AMD64 Architecture Programmer's Manual Vol 2 - System Programming.pdf
m. *(AMD)* AMD64 Architecture Processor Supplement - System V Application Binary Interface.pdf
n. reverse_engineering_the_microsoft_exfat_file_system.pdf
o. *(SUN MicroSystems)* ZFS On-Disk Specification.pdf
p. zfslast.pdf
q. Introduction to x64 Assembly.pdf
r. 03_Assembly_Language_Programming.ppt
s. asm_ch2_ia32.ppt
t. Computer Architecture for Esamus Students.ppt
u. Another Computer Model for ASM.pptx
v. multi-level paging.docx
w. Basic Memory Management (Without Swapping or Paging).docx
x. Swapping.docx