Odpovědi pište na zvláštní odpovědní list s vaším jménem a fotografií. Pokud budete odevzdávat více než jeden list s řešením, tak se na 2. a další listy nezapomeňte podepsat a do jejich záhlaví napsat i/N (kde i je číslo listu, N je celkový počet odevzdaných listů).

Otázka č. 1

Naimplementujte v Pascalu funkci `DelkaTextu` s následujícím prototypem:

```
type
  PUtf16 = ^word;
function DelkaTextu(text : PUtf16) : word;
```

která jako parametr `text` bere ukazatel na null-terminated řetězec v kódování UTF-16, a vrátí počet kompletních znaků (grafémů), ze kterých se tento řetězec skládá (tedy např. pro libovolný vstupní řetězec reprezentující text `Říp` má funkce `DelkaTextu` vrátit hodnotu 3). Předpokládejte, že velikost typu `word` jsou 2 byty. Pokud byste od RTL nutně potřebovali nějaké pomocné funkce nebo procedury, tak si je zadeklarujte, a popište chování, které od nich očekáváte. **Pozor:** vstupní řetězec `text` může být opravdu libovolný validní text v UTF-16 a nikoliv jen v UCS-2!

Otázka č. 2

Následující obrázek obsahuje část screenshotu hex editoru, který zobrazuje obsah 68 bytů dlouhého binárního souboru:

	0001	0203	0405	0607	0809	0A0B	0C0D	0E0F
00	0000	0000	0000	F07F	0000	0000	0000	F0FF
10	5000	5901	ED00	6C00	6900	6101	2000	7E01
20	6C00	7500	6501	6F00	7500	0D01	6B00	FD00
30	2000	6B00	6F01	4801	2E00	E000	80C7	7200
40	E100	6400						

Víme, že všechna data jsou v souboru uložena jako little endian, a že od 58. bytu (počítáno od 0) je v souboru uloženo 32-bitové reálné číslo s pohyblivou desetinnou čárkou. Mantisa je normalizována se skrytou 1 a zabírá spodních 23 bitů, pak následuje 8-bitový exponent uložený ve formátu s posunem (bias) +127 a 1 znaménkový bit. Zapište hodnotu tohoto reálného čísla v desítkové soustavě.

Otázka č. 3

Předpokládejte, že v OS s podporou pro vícevláknové zpracování dojde k náhlému ukončení nějakého vlákna (např. po dereferenci neplatného ukazatele) v situaci, kdy toto vlákno drží několik zamčených zámků. Implementace zámků je poskytována operačním systémem. Jak se v takové situaci má OS zachovat? Popište všechny typické možnosti řešení daného problému a vysvětlete jejich výhody a nevýhody.

Otázka č. 4

Předpokládejme, že v operačním systému poskytujícím podporu pro vícevláknové zpracování chceme naimplementovat proceduru `Sleep(s : Longint)`, která způsobí, že vlákno, které ji zavolá, nebude ve zpracování dalších instrukcí pokračovat dříve než za `s` sekund. Takovou operaci lze implementovat několika způsoby – vyberte pro zmíněný OS se souběžným během mnoha aplikací ten nejvhodnější, a v pseudokódu popište implementaci procedury `Sleep` a všech ostatních částí OS, které budou pro její fungování potřeba (soustřeďte se jen na části související se samotným procesem od začátku do konce čekání volajícího vlákna). Můžete počítat s tím, že každá cílová počítačová platforma vám poskytuje všechna běžná a pro vás důležitá zařízení a řadiče.

Otázka č. 5

Předpokládejte, že implementujete operační systém poskytující podporu pro vícevláknové zpracování a využívající mechanizmus stránkování pro oddělení adresových prostorů jednotlivých procesů běžících v systému, tj. váš OS vyžaduje pro svůj běh běžnou procesorovou platformu s podporou stránkování (dále uvažujte jen procesor s 32-bitovým fyzickým i virtuálním adresovým prostorem, a s jednoúrovňovými stránkovacími tabulkami, velikost jedné stránky si zvolte sami – svoji volbu explicitně uveďte a zdůvodněte jako součást vaší odpovědi). V takovém OS chceme pro každé nově vytvořené aplikační vlákno v adresovém prostoru vyhradit dostatek místa pro jeho zásobník (např. 4 MB). Zároveň bychom ale chtěli, aby skutečná fyzická paměť vyhrazená pro data zásobníku každého vlákna v každém okamžiku *přibližně* odpovídala maximu místa využitého na zásobníku daným vláknem od jeho spuštění (tj. po spuštění vlákna by jeho zásobník měl zabírat jen *malé* množství fyzické paměti, a pokud bude v průběhu svého života vlákno využívat větší část svého zásobníku, tak by se jím spotřebovaná fyzická paměť měla *postupně* zvětšovat). Velikost „*přibližně*“, „*malé*“ a „*postupně*“ vhodně zvolte a volbu zdůvodněte.
Je možné v popsaném kontextu takové chování nějak zařídit? Pokud ano, tak vysvětlete jak, a na příkladu popište v jakých situacích a jakým způsobem bude docházet ke zvětšování zásobníku (jím využité fyzické paměti). Pokud ne, tak vysvětlete proč.

Otázka č. 6

Předpokládejte, že chceme z USB 2.0 flash disku implementujícího protokol Mass Storage přečíst 256 bytů dat. Flash disk je přímo zapojený do kořenového hubu, který je součástí USB 2.0 řadiče v počítači (řadič implementuje EHCI). Flash disk je již plně nakonfigurovaný pro bulk přenos dat v plné rychlosti USB 2.0 (tzv. High Speed). Za předpokladu, že má náš flash disk na sběrnici USB přidělenou adresu ABCD, tak operace čtení přes Mass Storage vyžaduje:

1) Poslat 31 bytovou strukturu Command Block Wrapper (CBW) na adresu ABCD.
2) Přečíst z adresy ABCD 256 bytů vrácených dat.
3) Přečíst z adresy ABCD 13 bytů struktury Command Status Wrapper (CSW).

Popište, jaká (a jak strukturovaná) data se budou na nejnižší úrovni přenášet po sběrnici USB mezi řadičem a flash diskem v průběhu celé takové Mass Storage operace čtení. U každého bloku dat, kde to dává smysl, označte, kdo daná data po sběrnici USB posílá.

Otázka č. 7

Předpokládejte, že máme počítač IBM PC AT kompatibilní s procesorem Intel 80286 (16-bit procesor s 16-bit datovou a 24-bitovou adresovou sběrnicí, a zvláštním 16-bit I/O adresovým prostorem). Základní deska obsahuje dva 8-bitové ISA sloty (8 datových a 20 adresových vodičů [A0 až A19] + 4 vodiče pro rozlišení paměťové a I/O transakce, a čtení a zápisu [MEMR, MEMW, IOR, IOW]), a čtyři 16-bitové ISA sloty (navíc 8 datových a 4 adresové vodiče [A20 až A23]), a neobsahuje žádný řadič velkokapacitního úložného zařízení jako je pevný disk nebo disketová mechanika.
V základní desce je vloženo 8 SIPP paměťových modulů s celkovou kapacitou 2 MB DRAM. Ve 4. 16-bit ISA slotu je vložená 16-bit VGA grafická karta CL-GD5320 s 256 kB dedikované video RAM. V 1. 8-bit ISA slotu je vložena 8-bit síťová karta 3Com 3c503, která je připojená do 10 Mbit sítě Ethernet pomocí UTP kabelu. Okolní síťová infrastruktura je vhodně nakonfigurována.
Po zapnutí počítače dojde k nabootování operačního systému MS-DOS 6.22 ze sítě (stažením jádra OS z dostupného serveru). Popište, co se v počítači děje od jeho zapnutí do začátku stahování jádra MS-DOS. Popište hlavně to, jaké instrukce (a kterých programů) CPU po celou dobu startu počítače zpracovává, a odkud je přečte.

Otázka č. 8

Předpokládejte, že máte procesor, který má v sobě zabudovanou 8 MB velkou cache, a na kterém poběží váš program napsaný v Pascalu. Je potřeba při programování v Pascalu někdy znát velikost a princip fungování procesorové cache, nebo je její velikost a funkce pro běžné programování zcela irelevantní (tj. její velikost je třeba znát např. jen při programování přímo v assembleru, resp. strojovém kódu)? Vysvětlete proč!

Otázka č. 9

Předpokládejte, že máme diskový oddíl na disku s 64 B sektory naformátovaný souborovým systémem „unixového“ typu (podobný např. ext2 FS) s následujícím vlastnostmi:

- Každý inode má velikost 128 bytů, a má následující strukturu: pro identifikaci čísla sektoru se vždy používá plné 64-bitové číslo bez znaménka (první sektor v datové oblasti je označen číslem 1, hodnota 0 se v inodu používá pro značení nevyužitého odkazu na sektor), tabulka přímých odkazů na data souboru má 2 položky, následují 4 položky s nepřímými odkazy (postupně jedno, dvou, tří, a čtyř úrovňové).
- Každá adresářová položka vždy obsahuje 32-bitové číslo inodu, a právě 28 bytů jména souboru.

Předpokládejte, že je na disku vytvořený zcela prázdný adresář DIR1 (obsahuje 0 položek).
Za stavu, když jsou v souborovém systému datové sektory s číslem 7 a vyšším volné, vytvoříme postupně v adresáři DIR1 24 prázdných souborů (délka 0 bytů) pojmenovaných F1 až F24. Zapište obsah všech 6 položek odkazů na sektory u inodu DIR1 + obsah všech pomocných datových sektorů, které souborový systém po provedení všech výše uvedených operací používá pro správu dat samotného adresáře DIR1. Předpokládejte, že pokud je třeba v souborovém systému v jednom okamžiku alokovat více datových sektorů, tak se nejprve alokují pomocné sektory pro správu dat (a ty v pořadí, jak nastává jejich „logická“ potřeba), a až potom sektory pro samotná data souboru.

Otázka č. 10

Předpokládejte, že v počítači máme přes 32-bit sběrnici PCI připojený řadič GPIO (General Purpose Input/Output), který zpřístupňuje 32 nezávislých digitálních výstupních signálů/linek (tzv. GPIO). Řadič je nakonfigurovaný tak, že má od adresy `0x7708` v I/O adresovém prostoru namapovaný jeden 32-bitový port, kde každý z jeho bitů reprezentuje stav jednoho GPIO výstupního signálu. Čtením z tohoto portu zjistíme aktuální stav signálů, které řadič „vysílá“; zápisem nastavíme novou hodnotu signálů, které řadič „vysílá“. Vaším úkolem je v Pascalu s vhodným typickým rozšířením (např. Free Pascal) naimplementovat proceduru s následujícím prototypem:

```
procedure Output(b : Longword);
```

kde typ `Longword` je 32-bit celočíselný typ bez znaménka, a platná hodnota pro `b` je libovolné číslo `0` až `255`.
Účelem této procedury je **najednou (v jednom okamžiku)** změnit stav GPIO signálů „vysílaných“ GPIO řadičem následujícím způsobem (vše počítáno od `0`): 29. a 30. GPIO na `0`, 16. GPIO na `1`, 8. až 15. GPIO na hodnoty bitů 0 až 7 čísla `b`. Ostatní GPIO linky musí zůstat nezměněné!
Váš kód vždy poběží na počítači s procesorem Intel Pentium (64-bit datová sběrnice, 32-bit adresová sběrnice, separátní 16-bit I/O adresový prostor). Pořadí bitů i bytů chápané procesorem, sběrnicí PCI, i řadičem GPIO jsou stejná.
Procesor má mimo jiné 4 obecné 32-bitové registry EAX, EBX, ECX, EDX. Instrukční sada obsahuje mimo jiné následující instrukce:

- `MOV` *op1*`,` *op2*
  Kde jen jeden z *op1* a *op2* může být adresa, *op1* = cíl (registr nebo adresa), *op2* = zdroj (registr, adresa [označená hranatými závorkami] nebo hodnota immediate).
- `IN EAX, EDX`
  Přečtení 32-bitové hodnoty do EAX z adresy (v EDX) z I/O adresového prostoru.
- `OUT EDX, EAX`
  Zápis hodnoty EAX do 32-bitů v I/O adresovém prostoru od adresy dané EDX.

Dále jsou v instrukční sadě obsaženy instrukce pro všechny běžné unární i binární aritmetické a bitové operace – takové instrukce mají podobu: `unInstr` *op1* nebo `binIntr` *op1*`,` *op2*, kde: *op1* = cíl (pouze registr), *op2* = 2. zdroj (registr, immediate, nebo adresa).
Pozor: instrukce `IN` a `OUT` mají u procesoru Intel Pentium i (výše neuvedené) varianty s **8-bit, resp. 16-bit** operandem, které zde ale **nesmíme** použít, přestože na první pohled vypadá jejich použití jako ekvivalentní – protože např. čtyři postupné 8-bitové zápisy na adresy `0x7708`, `0x7709`, `0x770A`, `0x770B` nejsou pro nás ekvivalentní jednomu 32-bitovému zápisu na adresu `0x7708`, protože **způsobí jen postupnou změnu** potřebných linek GPIO.