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

Předpokládejte, že máme čistě naformátovaný souborový systém používající tabulku FAT – kde, jeden záznam ve FAT má 8 bitů, hodnota 0 reprezentuje volný sektor, konec souboru reprezentuje maximální hodnota. Souborový systém je na disku se 128 B sektory, velikost jednoho clusteru je 1 sektor. První sektor/cluster použitelný pro data souborů (tj. 1. datový sektor) je označený číslem 1, a odpovídá mu první záznam ve FAT. Souborový systém podporuje pouze jeden adresář (kořenový), a pro jeho obsah jsou napevno vyhrazeny 3 sektory před 1. datovým sektorem. Velikost jedné adresářové položky je 32 B. Každá adresářová položka obsahuje mimo jiné 11 B pro jméno souboru, 1 B pro číslo prvního sektoru, a 2 B pro velikost souboru v bytech.

Nakreslete konečný obsah prvních 16 záznamů FAT tabulky po provedení následujících operací (předpokládejte, že pokud je potřeba další volný sektor pro data souboru, tak se v souborovém systému vybere první volný sektor s nejnižším číslem; operace „zápis N bytů do X“ se chápe jako připsání N bytů za poslední byte souboru X):

1) Vytvoření prázdného souboru A.TXT
2) Vytvoření prázdného souboru B.TXT
3) Zápis 200 bytů do B.TXT
4) Zápis 50 bytů do B.TXT
5) Zápis 1 kB do A.TXT
6) Zápis 150 bytů do B.TXT
7) Vytvoření prázdného souboru C.TXT
8) Zápis 1 bytu do C.TXT

Otázka č. 2

Předpokládejte, že máme 3 jednočipové počítače A, B, a C propojené jednou sběrnicí I^2C. Všechna připojená zařízení používají rychlost 100 kb/s, a 7 bitové adresování. Zařízení mají nastaveny následující adresy: A = 09h, B = 07h, C = 7Ah. Předpokládejte, že na sběrnici nikdo nevysílá, a zařízení C se rozhodne poslat zařízení A následující 2 byty: FF 55. Nakreslete a popište průběh napětí (a jejich logických hodnot) na vodičích Serial Data Line (SDA) and Serial Clock (SCL) od začátku až do konce výše zmíněného přenosu (předpokládejte, že během přenosu nedojde k žádným chybám).

Otázka č. 3

Napište program v jazyce Pascal (případně v jazyce C), který na standardní výstup (tj. pomocí procedury `WriteLn`) vypíše text „`Little Endian`“ nebo „`Big Endian`“ bez uvozovek podle toho, na jaké platformě bude spuštěn (resp. pro kterou bude přeložen). Připomenutí: prefixový unární operátor @ slouží v Pascalu pro získání adresy libovolné proměnné.

Otázka č. 4

Spočítejte hodnotu následujícího výrazu zapsaného v Pascalu (předpokládejte, že celý výpočet i všechny uvedené hodnoty jsou v 64-bitových celých číslech bez znaménka):

```
(10 OR $2000) XOR ($3 SHL 1)
```

Výsledek zapište jako jedno celé číslo v desítkové soustavě.

Otázka č. 5

Předpokládejte, že implementujete funkci OS, která pošle N bytů dat po lokální síti. Funkce OS dostane od aplikace v jednom parametru ukazatel na první byte, který má po síti poslat, a ve druhém parametru číslo N. Pro komunikaci se síťovou kartou (pro zápis do jejích registrů) se používá mechanizmus *port-mapped IO*, a síťová karta používá mechanizmus DMA pro přenos dat z/do hlavní paměti RAM. Před začátkem zápisu tedy musí OS do registrů síťové karty zapsat zdrojovou adresu v paměti RAM, která ukazuje na data připravená pro odeslání. Kvůli optimalizaci výkonu bychom ale chtěli podporovat jen situaci, kdy síťová karta vždy získá přímo adresu původních dat ve zdrojové aplikaci (tj. před provedením operace odeslání nikdy nechceme data kopírovat na jiné místo v paměti). Za předpokladu, že se v OS používá mechanizmus stránkování, vysvětlete následující:

a) Vysvětlete, co vše musí OS v takové situaci udělat, aby síťové kartě předal správnou adresu. Na příkladu uveďte jakou.
b) Bude tento princip fungovat pro všechny možné adresy a hodnoty N, které může aplikace operačnímu systému předat? Vysvětlete proč ano, resp. proč ne.

Otázka č. 6

Následující obrázek obsahuje část screenshotu hex editoru, který zobrazuje obsah 52 bytů dlouhého binárního souboru:

```
    0001 0203 0405 0607 0809 0A0B 0C0D 0E0F
00  0000 0000 0000 F07F 0000 0000 0000 F0FF
10  7200 E100 6400 0000 67C1 5801 ED00 7000
20  2000 6A00 6500 2000 6B00 6F00 7000 6500
30  6300 2E00
```

Víme, že všechna data jsou v souboru uložena jako little endian, a že od 22. 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. 7

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. 8

Předpokládejme následující část programu v jazyce Pascal (jednotlivé řádky programu v Pascalu jsou očíslované a označené *kurzívou*; pod každým řádkem v Pascalu jsou vypsané instrukce procesorové řady x86, na prvním řádku jsou vždy zapsané byty strojového kódu dané instrukce, na druhém řádku je pak v odsazení uveden zápis dané instrukce v Intel assembleru; proměnné `a`, `b`, `c` jsou typu `Longint`):

```
ř15: a := a + b;
      A1 20 C0 40 00
            mov    eax, [0040C020h]
      8B 15 30 C0 40 00
            mov    edx, [0040C030h]
      01 D0
            add    eax, edx
      A3 20 C0 40 00
            mov    [0040C020h], eax
ř16: b := 0;
      C7 05 30 C0 40 00 00 00 00 00
            mov   [0040C030h], 0
ř17: c := a + 8;
      A1 20 C0 40 00
            mov    eax, [0040C020h]
      83 C0 08
            add    eax, 8
      A3 40 C0 40 00
            mov    [0040C040h], eax
```

Nyní předpokládejme, že chceme tento program ladit a na řádek číslo 17 umístit breakpoint. V tomto kontextu odpovězte na následující otázky:

a) Je třeba, aby debugger rozuměl zdrojovým kódům jazyka Pascal? A pokud ne, jak debugger pozná, že má vykonávání programu zastavit zrovna před provedením instrukce `mov eax, [0040C020h]`?
b) Jak debugger způsobí, že se program „zastaví“ před provedením instrukce `mov eax, [0040C020h]`, když přeci procesor stále musí nějaký kód vykonávat?

Otázka č. 9

Následující program zapsaný v jazyce Pascal můžete pomocí překladače Free Pascal přeložit a spustit jak na OS Linux na platformě x86, tak i na OS Windows 8 na platformě x86. Popište a vysvětlete, z jakého důvodu je to možné a jaké všechny kroky je třeba provést, abyste z původního zdrojového souboru získali spustitelný soubor. Do výkladu zahrňte zdůvodnění, zda a proč pro daný scénář bude potřeba více různých spustitelných souborů daného programu, nebo zda a proč bude dostačovat pouze jeden.

```
program Mocnina;
var
  m, n : integer;

begin
  ReadLn(n);
  m := 1;
  while n >= 1 do begin
    m := m * 2;
    Dec(n);
  end;
  WriteLn('2 na N je ', m);
end.
```

Otázka č. 10

Naimplementujte v Pascalu funkci `DelkaTextu` s následujícím prototypem:

```
type
  PUtf32 = ^longword;

function DelkaTextu(text : PUtf32)
         : longword;
```

která jako parametr `text` bere ukazatel na null-terminated řetězec v kódování UTF-32, 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 `longword` jsou 4 byty). Pokud byste od RTL nutně potřebovali nějakou pomocnou funkci nebo proceduru, tak si ji zadeklarujte, a popište chování, které od ní očekáváte.

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

Mikroprocesor MOC 6502 je 8-bitový procesor se 16-bitovou adresovou sběrnicí (paměťovým adresovým prostorem) a s akumulátorovou architekturou. Registr akumulátoru má v assembleru jméno A. V příznakovém registru je příznak Carry (přenos) se standardním chováním označený písmenem C.

Procesor 6502 má následující instrukce:

- LDA `$`*`adresa`*`/#`*`konstanta`* (Load Accumulator) – načtení 8-bitové hodnoty ze zadané 16-bitové adresy (argument instrukce uvozený `$`) *nebo* přímo zadané 8-bitové konstanty (argument instrukce uvozený #) do akumulátoru.
- STA `$`*`adresa`* (Store Accumulator) – uložení hodnoty akumulátoru na zadanou 16-bitovou adresu (argument instrukce)
- ADC `$`*`adresa`*`/#`*`konstanta`* (Add with Carry) – sečtení dvou 8-bitových čísel a příznaku C (druhý operand operace sčítání je argumentem instrukce)
- SBC `$`*`adresa`*`/#`*`konstanta`* (Subtract with Carry) – odečtení dvou 8-bitových čísel a negace příznaku C (druhý operand operace odečítání je argumentem instrukce)
- CLC (Clear Carry) – nastavení příznaku C na 0 (instrukce bez explicitních argumentů)
- SEC (Set Carry) – nastavení příznaku C na 1 (instrukce bez explicitních argumentů)

Přepište následující výraz v Pascalu do ekvivalentní posloupnosti instrukcí strojového kódu procesoru 6502 (16-bitové proměnné E, F v sobě obsahují celá bezznaménková čísla a jsou uloženy na následujících adresách: E = E002h, F = F002h). Všechny operace chceme provést v celých číslech s 16-bitovou přesností bez znaménka (všechny hodnoty jsou Little Endian):

```
E := E + F + 16
```

Otázka č. 2

V kódování Unicode existuje následující přiřazení kódů jednotlivým znakům: kód 158h pro znak ‚Ř', kód 52h pro znak ‚R', kód 65h pro znak ‚e', kód 70h pro znak ‚p', kód 61h pro znak ‚a', a kód 30Ch pro kombinující znak (combining character) diakritické znaménko háček (stejné znaménko jaké je použito u znaku ‚Ř').

a) Je v kódování Unicode nějaký významový rozdíl mezi znakem 158h a posloupností znaků 52h 30Ch? Pokud ano, tak jaký?

b) Od adresy 0 chceme do paměti uložit text „Řepa" (bez uvozovek) v kódování UTF-16 ve variantě Little Endian. V šestnáctkové soustavě zapište hodnoty jednotlivých bytů paměti od adresy 0, které v sobě budou obsahovat část výše uvedeného textu v daném kódování.

Otázka č. 3

Následující hodnotu:

-256,625

zapište jako 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. Výsledek zapište jako hodnoty jednotlivých bitů v pořadí zleva doprava jako MSB first.

Otázka č. 4

Mějme 8-bitový jednočipový počítač Intel 8051 s akumulátorovou architekturou a následující instrukční sadou:

- `MOV A, #`*`konstanta`* – načtení 8-bitové konstanty do akumulátoru
- `ADD A, `*`adresa`* – přičtení (bez přenosu) 8-bitové hodnoty uložené na 8-bitové adrese *adresa*

V interní paměti našeho MCU 8051 je od adresy 0 uložen následující program ve strojovém kódu 8051 (každý sudý řádek obsahuje přepis instrukce do assembleru 8051):

```
0x74 0x05
    MOV A, #5
0x25 0x03
    ADD A, 3
0x25 0x04
    ADD A, 4
```

Předpokládejte, že počítač začne tento program provádět (např. jako následek nepodmíněného skoku na adresu 0). Rozhodněte a zdůvodněte, zda můžeme určit obsah registru A po provedení všech instrukcí výše uvedeného programu, a pokud ano, tak jaká hodnota A je? **Pozor:** procesor 8051 má čistou Harvardskou architekturu.

Otázka č. 5

Předpokládejte, že máme 3 jednočipové počítače A, B, a C propojené jednou sběrnicí I^2C. Všechna připojená zařízení používají rychlost 100 kb/s, a 7 bitové adresování. Zařízení mají nastaveny následující adresy: A = 02h, B = 07h, C = A1h. Předpokládejte, že na sběrnici nikdo nevysílá, a zařízení A se rozhodne poslat zařízení B následující 2 byty: AA 0F. Nakreslete a popište průběh napětí (a jejich logických hodnot) na vodičích Serial Data (SDA) a Serial Clock (SCL) od začátku až do konce výše zmíněného přenosu (předpokládejte, že během přenosu nedojde k žádným chybám).

Otázka č. 6

Popište všechny typické stavy, ve kterých může být vlákno v běžném systému s vícevláknovým zpracováním. Popište také všechny běžné přechody mezi jednotlivými stavy vlákna a vysvětlete, v jaké situaci k danému přechodu dochází.

Otázka č. 7

Předpokládejte, že máme počítač IBM PC kompatibilní s procesorem Intel 80386DX (32-bit procesor s 32-bit datovou a 32-bitovou adresovou sběrnicí, a zvláštním 16-bit I/O adresovým prostorem). Základní deska obsahuje pouze 16-bitové ISA sloty (16 datových a 24 adresových vodičů + vodič pro rozlišení paměťové a I/O transakce [M/IO]), a neobsahuje žádný řadič velkokapacitního úložného zařízení jako je pevný disk nebo disketová mechanika.

V základní desce jsou vloženy 2 SIMM paměťové moduly s celkovou kapacitou 2 MB DRAM. V 1. ISA slotu je vložená 16-bit VGA grafická karta Trident 8900C s 512 kB dedikované video RAM. Ve 3. ISA slotu je vložena 16-bit síťová karta 3Com 3c509C, 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 5.0 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 chcete programovat větší množství různých aplikací, které všechny budou používat grafické uživatelské rozhraní (GUI). Každou z naprogramovaných aplikací budete chtít v binární spustitelné podobě distribuovat pro operační systémy Linux a Mac OS X. Oba tyto operační systémy mají ale rozdílné API (navzájem nekompatibilní) pro tvorbu grafického uživatelského rozhraní, a zároveň programovací jazyk, který budete používat pro programování aplikací, nemá žádnou podporu pro tvorbu GUI. Aplikace chcete programovat tak, aby byly přenositelné na úrovni zdrojového kódu mezi oběma uvedenými systémy. Jakým způsobem to nejlépe zařídíte? Jakým způsobem pak bude probíhat překlad veškerého potřebného kódu každé takové aplikace, až do stavu, kdy máme finální spustitelné soubory?

Otázka č. 9
(otázka za celkem 2 body)

Předpokládejte počítač s 32-bitovým paměťovým adresovým prostorem. V systému je nainstalována 64 kB velká paměť ROM, která je souvisle namapována na nejvyšší možné adresy v paměťovém adresovém prostoru počítače. Dále je v systému nainstalováno 0,25 GB paměti RAM, která je souvisle namapována od adresy `0` v paměťovém adresovém prostoru počítače, s výjimkou adres `1F1h` až `178h` na kterých jsou namapovány porty HDC pomocí mechanizmu MM I/O. Předpokládejte, že v Pascalu (případně v jazyce C) implementujete část firmware počítače, který bude uložený ve zmíněné paměti ROM. Napište implementaci funkce `Write` (a případně dalších procedur a funkcí, které budete potřebovat) s následujícím prototypem (viz níže), která má zařídit zapsání 1 sektoru dat (vždy `512` bytů) na pevný disk – parametr `sector` určuje číslo sektoru na disku, který se má zapsat; parametr `data` obsahuje ukazatel na `512` bytů dat v paměti, které se mají zapsat do daného sektoru (ukazatel samozřejmě obsahuje adresu 1. z `512` bytů). Funkce se vrátí ihned, jak to bude možné a nečeká na kompletní dokončení operace zápisu (tj. je asynchronní operací zápisu na pevný disk). Funkce vrací: (a) `True`, pokud došlo k úspěšnému spuštění operace zápisu, (b) `False`, pokud ještě probíhá předchozí operace zápisu na pevný disk a tedy požadovaný zápis není možné provést:

```
type
      PByte = ^Byte;
function Write(sector : Word;
               data : PByte) : Boolean;
procedure InitHarddisk;
procedure SetInterruptVector(
               intVec : Integer;
               handlerRoutine : Pointer);
```

Při inicializaci firmware počítače se mimo jiné volá i vaše procedura `InitHarddisk` (viz výše), do které můžete naimplementovat libovolnou inicializaci (např. vašich globálních proměnných), kterou budete potřebovat. Dále je vám k dispozici předpřipravená procedura `SetInterruptVector`, která do vektoru přerušení s číslem `intVec` nastaví adresu obslužné procedury předané v parametru `handlerRoutine`. Můžete očekávat, že od začátku do konce běhu obslužné procedury jsou zakázána všechna přerušení.

S řadičem pevného disku se komunikuje pomocí mechanizmu PIO. Pro iniciaci operace zápisu na pevný disk je třeba provést následující posloupnost zápisů do portů jeho řadiče (vždy je uvedena adresa, na které je daný port namapovaný):

1) `1F6h`: horních 8 bitů čísla sektoru
2) `1F4h`: spodních 8 bitů čísla sektoru
3) `1F8h`: 8-bitový identifikátor příkazu: příkaz `WRITE SECTOR` = hodnota `80h`

Poté je třeba vyčkat, až bude řadič připraven na přijímání dat – to řadič indikuje nastavením 6. bitu (číslované od 0), tzv. `BSY` (Busy), ve stavovém portu na adrese `1F8h` na hodnotu `0`. Poté je možné zapisovat jednotlivé byty (které mají být zapsány do vybraného sektoru) do datového portu na adrese `1F1h`. Zápis každého bytu do datového portu způsobí nastavení bitu `BSY` na hodnotu `1`. Před zápisem každého dalšího datového bytu je tedy třeba vždy vyčkat, až řadič opět oznámí svoji připravenost nastavením `BSY` na `0`. Každé nastavení bitu `BSY` na hodnotu `0` je řadičem disku indikováno vyvoláním přerušení číslo `14`. Pozor: v obsluze přerušení `14` je třeba ověřit, že přerušení opravdu vyvolal řadič disků a ne jiný zdroj, tj. že hodnota bitu `BSY` je opravdu `0`. Pokud přerušení pochází z jiného zdroje, je možné ho ignorovat. Předpokládejte, že není zapnutá segmentace, ani stránkování. Předpokládejte, že typ `Word` slouží pro ukládání bezznaménkových celých čísel a jeho velikost je 16-bitů.

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.

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

Předpokládejte systém s preemtivním plánováním vláken, kde každé vlákno má přidělenou nějakou pevnou prioritu v rozsahu 0-31 (0 je nejvyšší priorita, 31 je nejnižší priorita). Systém aplikacím též poskytuje standardní implementaci zámků (mutexů). Předpokládejte, že na takovém systému spustíme 2 vlákna, která provádí následující posloupnosti operací (předpokládejte, že `lock X` je zamčení zámku *X*, `unlock X` je odemčení zámku *X*, `opShort` je nějaká krátce trvající operace [trvá v řádu jednotek taktů procesoru], `opLong` je nějaká dlouho trvající operace [trvá v řádu jednotek sekund]):

vlákno *T1* (priorita 14):

```
lock A
opShort
unlock A
lock B
opLong
lock A
opLong
unlock A
opLong
unlock B
```

vlákno *T2* (priorita 15):

```
lock C
lock A
lock B
opShort
unlock C
unlock B
unlock A
```

V tomto kontextu odpovězte na následující: Může někdy dojít k problému zvanému *deadlock*? Pokud ano, popište alespoň jeden takový případ. Pokud ne, vysvětlete proč.

Otázka č. 2

Naimplementujte v Pascalu funkci `DelkaTextu` s následujícím prototypem:

```
type
  PUtf8 = ^byte;
function DelkaTextu(text : PUtf8) : word;
```

která jako parametr `text` bere ukazatel na null-terminated řetězec v kódování UTF-8, 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, velikost typu `byte` je 1 byte. 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-8!

Otázka č. 3

CIL kód je strojový kód virtuálního stroje se zásobníkovou architekturou (zásobníkového stroje). CIL kód má load/store architekturu. CIL kód má následující instrukce:

- `LDSFLD` *`adresa`* – načtení hodnoty na zadané adrese (argument instrukce)
- `STSFLD` *`adresa`* – uložení hodnoty na zadanou adresu (argument instrukce)
- `LDC` *`číslo`* – načtení celočíselné konstanty (argument instrukce)
- `ADD` – sečtení dvou čísel (instrukce bez explicitních argumentů)
- `MUL` – vynásobení dvou čísel (instrukce bez explicitních argumentů)

Předpokládejte, že registrový zásobník má neomezenou hloubku. Přepište následující výraz v Pascalu do ekvivalentní posloupnosti instrukcí strojového kódu CIL (proměnné A, B, C jsou uloženy na následujících adresách: A = `$5000`, B = `$6000`, C = `$7000`):

```
A := (A * B) + C + (16 * A)
```

Otázka č. 4

Předpokládejte programové prostředí, které poskytuje podporu pro vícevláknové zpracování. Předpokládejte, že hlavní procedura nějakého vlákna doběhne do konce (tj. toto vlákno je ve stavu těsně před provedením instrukce RET [instrukce návratu z podprogramu] na konci jeho hlavní procedury). Popište a vysvětlete, co se bude dít potom (jaký kód bude procesor dále zpracovávat a co tento kód bude dělat).

Otázka č. 5

Předpokládejte, že máme aplikaci A, která pro komunikaci s OS používá POSIX API verze 1b. Aplikaci A máme přeloženou tak, že výsledný binární spustitelný soubor XX jsme schopni bez problémů spustit na běžné instalaci OS Linux (který též implementuje POSIX.1b API). Dále víme, že existuje SW abstrakční vrstva Cygwin nad Win32 API pro Windows řady NT, a máme verzi Cygwinu, který implementuje všechny procedury a funkce z POSIX.1b API, které používá aplikace A. Jsou výše uvedené informace dostatečné pro to, abychom rozhodli, zda jsme pomocí Cygwin schopni spustit binární spustitelný soubor XX na Windows 7 (řada NT)? Pokud ano, vysvětlete proč. Pokud ne, napište všechny informace, které nám pro rozhodnutí schází, a vysvětlete proč.

Společná část pro otázky označené X

Předpokládejte, že máme osobní počítač s následující architekturou:
① Hlavní systémová sběrnice je paralelní 66 MHz sběrnice s 36-bit adresovou a dedikovanou 64-bit datovou částí. Na systémovou sběrnici jsou připojeny 2 CPU Intel Pentium Pro (32-bit procesor s instrukční sadou x86, s 36-bit paměťovým adresovým prostorem, 16-bit I/O adresovým prostorem, 64-bit datovou sběrnicí; taktovací frekvence jádra je 166 MHz, součástí procesoru je 512 kB L2 cache, a 32 kB L1 cache, obě cache mají velikost řádky 32 B).
② Třetím zařízením připojeným na systémovou sběrnici jsou čipy Intel 82441FX „PCI and Memory Controller“ (PMC) a Intel 82442FX „Data Bus Accelerator“ (DBX), které dohromady tvoří northbridge chipsetu Intel 440FX. Northbridge je připojen na všechny datové vodiče systémové sběrnice, ale jen na adresové vodiče HA3 až HA31 (počítáno od 0). Northbridge v sobě obsahuje řadič paměti pro SIMM paměťové moduly DRAM – paměťová sběrnice je paralelní s 64-bit datovou a dedikovanou 30-bit adresovou částí). Na základní desce je v SIMM modulech nainstalován plný 1 GB maximální northbridgem podporované kapacity DRAM. Součástí northbridge je též PCI host bridge pro paralelní 33 MHz 32-bit sběrnici PCI (sdílená 32-bit adresová a datová sběrnice). Sběrnice PCI má oddělený paměťový a I/O adresový prostor.
③ Na sběrnici PCI je připojen southbridge Intel 82371SB „PCI I/O IDE Xcelerator“ (PIIX3). Stejná PCI sběrnice je též vyvedena do 4 PCI slotů na základní desce. PIIX3 v sobě mimo jiné obsahuje PCI-ISA bridge pro 16-bit ISA sběrnici (paralelní 8 MHz sběrnice, 16-bit datová a dedikovaná 24-bit adresová sběrnice, zvláštní 16-bit I/O adresový prostor). Tato ISA sběrnice je vyvedena do 2 ISA slotů na základní desce.
④ V 1. PCI slotu je vložena grafická karta S3 Trio V64. Ve 2. PCI slotu je vložena SCSI [čti “skazi”] karta Adaptec AHA-2940 (s HBA [řadičem] AIC-7850). Ke kartě je po sběrnici SCSI (1. SCSI sběrnice v počítači) připojen pevný disk HDA s jednou partition s FAT32 FS, a pevný disk HDB s jednou partition s ext2 FS. V 1. ISA slotu je vložena SCSI karta Adaptec AVA-1505 (s HBA [řadičem] AIC-6260). Ke kartě je po SCSI sběrnici (2. SCSI sběrnice v počítači) připojen scanner HP ScanJet 4P. Řadiče AIC-7850 a AIC-6260 mají rozdílné vzájemně nekompatibilní HCI.

SCSI je paralelní multidrop sběrnice. Po sběrnici SCSI se data přenášejí ve formě paketů. Formát SCSI paketů je standardizovaný. Payload každého takového paketu tvoří příkazy připojeným zařízením a odpovědi na ně. Pro každý typ zařízení připojitelného na sběrnici SCSI (disk nebo CD-ROM mechanika nebo scanner) existuje standardizovaná množina a formát takových příkazů.

Otázka č. 6 (X)

Vysvětlete, jaké jsou pravděpodobné důvody k tomu, že je uvedený northbridge připojený jen k adresovým vodičům 3 až 31 na systémové sběrnici.

Otázka č. 7 (X)

Pro dané PC bychom chtěli vyrobit PCI kartu s N GB přídavné paměti DRAM tak, aby tato paměť byla pro aplikace přístupná podobně jako 1 GB hlavní paměti RAM (a její použití bylo pro aplikace maximálně transparentní). Pro obě následující varianty rozhodněte a vysvětlete, zda by to bylo možné, a popište, jak by v takové variantě aplikace přistoupila např. k prvnímu a jak např. k poslednímu bajtu přídavné paměti RAM:

a) N = 2 GB
b) N = 8 GB

Otázka č. 8 (X)

Předpokládejte, že implementujeme jádro nového OS, který má běžet na výše uvedeném PC, a má podporovat práci s veškerými instalovanými zařízeními. Navrhněte (nakreslete obrázek) jakým způsobem bychom v takové situaci jádro OS strukturovali. Zaměřte se na podporu aplikačního souborového API, a aplikačního API pro čtení obrázku ze scannerů minimálně ve výše uvedené situaci. Pokud budete kód jádra OS rozdělovat do více modulů, tak každý takový modul označte a stručně popište jeho funkci, a vyznačte všechny ostatní moduly a části OS, se kterými bude každý takový modul komunikovat při běžném požadavku od aplikace (např. čtení nebo zápis X bytů dat z/do nějakého souboru, čtení Y bytů obrazových dat ze scanneru). Dále bychom chtěli, aby náš OS byl jednoduše na úrovni zdrojových kódů přenositelný i na jinou procesorovou architekturu než x86.

Otázka č. 9

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 33. bytu (počítáno od 0) je v souboru uloženo 16-bitové reálné číslo s pevnou desetinnou čárkou ve formátu 5+11. Zapište hodnotu tohoto reálného čísla v desítkové soustavě.

Otázka č. 10

Předpokládejte provedení následující instrukce:

```
MOV   EAX, [2046]
```

která načítá 4 bytovou hodnotu z adresy 2046 do registru EAX. Operační kód instrukce začíná na adrese 0xFFFE a její celková délka je 6 bytů. Systém používá stránky o velikosti 1 kB a jednoúrovňové stránkovací tabulky. Rozhodněte, kolik výpadků stránky (page faults) může celkem maximálně vzniknout v průběhu zpracování této instrukce. Popište proč. Očekávejte, že při každém výpadku stránky dojde k obnovení mapování dané stránky a v rámci zpracování dané instrukce již k dalšímu výpadku stejné stránky nedojde.

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

Předpokládejte, že máme v Pascalu funkci s následujícím prototypem:

```
function NaFixedPoint(
      hodnota : longint; exp : longint
) : longint;
```

Parametry funkce reprezentují reálné číslo `x` následujícím způsobem: `x = hodnota * 10`^{exp}. Funkce vrací hodnotu `x` jako reálné číslo s pevnou desetinou čárkou ve formátu 10+22. V šestnáctkové soustavě zapište návratovou hodnotu funkce `NaFixedPoint` po zavolání s následujícími parametry (reprezentujícími hodnotu `11,53125`):

```
… := NaFixedPoint(1153125, -5);
```

Předpokládejte, že typ `longint` je běžná Pascalová reprezentace celého 32-bitového znaménkového (signed) čísla.

Otázka č. 2

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. 3

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. 4

Předpokládejte, že máme čistě naformátovaný souborový systém používající tabulku FAT – kde, jeden záznam ve FAT má 12 bitů, hodnota 0 reprezentuje volný sektor, konec souboru reprezentuje maximální hodnota. Souborový systém je na disku s 512 B sektory, velikost jednoho clusteru je 1 sektor. První sektor/cluster použitelný pro data souborů (tj. 1. datový sektor) je označený číslem 1, a odpovídá mu první záznam ve FAT. Souborový systém podporuje pouze jeden adresář (kořenový), a pro jeho obsah jsou napevno vyhrazeny 4 sektory před 1. datovým sektorem. Velikost jedné adresářové položky je 32 B. Každá adresářová položka obsahuje mimo jiné 11 B pro jméno souboru, 2 B pro číslo prvního sektoru, a 4 B pro velikost souboru v bytech.

Nakreslete konečný obsah prvních 16 záznamů FAT tabulky po provedení následujících operací (předpokládejte, že pokud je potřeba další volný sektor pro data souboru, tak se v souborovém systému vybere první volný sektor s nejnižším číslem; operace „zápis N bytů do X“ se chápe jako připsání N bytů za poslední byte souboru X):

1) Vytvoření prázdného souboru A.TXT
2) Vytvoření prázdného souboru B.TXT
3) Zápis 500 bytů do B.TXT
4) Zápis 4 KiB do A.TXT
5) Zápis 1 KiB do B.TXT
6) Smazání celého souboru A.TXT
7) Vytvoření prázdného souboru C.TXT
8) Zápis 1 bytu do B.TXT
9) Zápis 512 bytů do B.TXT

Otázka č. 5

Následující program zapsaný v jazyce Pascal můžete pomocí překladače Free Pascal přeložit a spustit jak na OS Linux na platformě x86, tak i na OS Windows XP na platformě x86. Popište a vysvětlete, z jakého důvodu je to možné a jaké všechny kroky je třeba provést, abyste z původního zdrojového souboru získali spustitelný soubor. Do výkladu zahrňte zdůvodnění, zda a proč pro daný scénář bude potřeba více různých spustitelných souborů daného programu, nebo zda a proč bude dostačovat pouze jeden.

```
program Mocnina;
var
  m, n : integer;

begin
  ReadLn(n);
  m := 1;
  while n >= 1 do begin
    m := m * 2;
    Dec(n);
  end;
  WriteLn('2 na N je ', m);
end.
```

Otázka č. 6

Předpokládejte, že implementujete funkci OS, která pošle N bytů dat po lokální síti. Funkce OS dostane od aplikace v jednom parametru ukazatel na první byte, který má po síti poslat, a ve druhém parametru číslo N. Pro komunikaci se síťovou kartou (pro zápis do jejích registrů) se používá mechanizmus *port-mapped IO*, a síťová karta používá mechanizmus DMA s podporou *scatter/gather IO* pro přenos dat z/do hlavní paměti RAM. Před začátkem zápisu tedy musí OS síťové kartě předat zdrojovou adresu v paměti RAM, která ukazuje na data připravená pro odeslání. Kvůli optimalizaci výkonu bychom ale chtěli podporovat jen situaci, kdy síťová karta vždy získá přímo adresu původních dat ve zdrojové aplikaci (tj. před provedením operace odeslání nikdy nechceme data kopírovat na jiné místo v paměti). Za předpokladu, že se v **OS používá mechanizmus stránkování**, vysvětlete následující:

a) Vysvětlete, co vše musí OS v takové situaci udělat, aby síťové kartě předal správnou adresu. Na příkladu uveďte jakou.
b) Bude tento princip fungovat pro všechny možné adresy a hodnoty N, které může aplikace operačnímu systému předat? Vysvětlete proč (při jakých kombinacích adresy a N) ano, resp. proč (a kdy) ne.

Otázka č. 7

Předpokládejte, že navrhujete zvukovou kartu pro 16-bitovou sběrnici ISA (16-bit datová sběrnice, dedikovaná 24-bitová adresová sběrnice, oddělený 16-bitový I/O adresový prostor, podpora pro bus mastering). Zvuková karta má podporovat pouze přehrávání nekomprimovaného zvuku pouze v CD formátu (2 kanálový zvuk [stereo], 16-bitové vzorky, vzorkovací frekvence 44,1 kHz). Navrhněte HCI takové zvukové karty (popište všechny jeho části; předpokládejte, že adresy portů a veškeré další konstanty si můžete zvolit libovolně smysluplně), které bude splňovat následující požadavky:

1) Podporuje scatter/gather IO.
2) Jedním příkazem „Play“ lze spustit přehrávání audia (parametrem příkazu je buffer se zvukem, a velikost bufferu v bytech [podporujte smysluplnou maximální velikost bufferu]).
3) Dokud karta přenáší nějaká data z operační paměti počítače, tak tento stav vhodně indikuje, a nepřijímá (= ignoruje) další příkazy „Play“.

Otázka č. 8

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. 9

Popište, co znamená pojem *sandbox*, a vysvětlete, jak daný koncept funguje a k čemu se používá.

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 `0x4567` 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`): 2. a 3. GPIO na `0`, 20. až 23. GPIO na hodnoty bitů 0 až 3 čísla `b`, 28. až 31. GPIO na **negaci** hodnot bitů 4 až 7 čísla `b`. Ostatní GPIO linky musí zůstat nezměněné!
Váš kód vždy poběží na počítači s procesorem Intel Pentium III (32-bitový procesor, 64-bit datová sběrnice, 36-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 III 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 `0x4567`, `0x4568`, `0x4569`, `0x456A` nejsou pro nás ekvivalentní jednomu 32-bitovému zápisu na adresu `0x4567`, protože **způsobí jen postupnou změnu** potřebných linek GPIO.

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).

Společná část pro otázky označené X

Předpokládejte, že máme osobní počítač s následující architekturou:

① Hlavní systémová sběrnice je paralelní 66 MHz sběrnice s 36-bit adresovou a dedikovanou 64-bit datovou částí. Na systémovou sběrnici jsou připojeny 2 CPU Intel Pentium Pro (32-bit procesor s instrukční sadou x86, s 36-bit paměťovým adresovým prostorem, 16-bit I/O adresovým prostorem, 64-bit datovou sběrnicí; taktovací frekvence jádra je 166 MHz, součástí procesoru je 512 kB L2 cache, a 32 kB L1 cache, obě cache mají velikost řádky 32 B).

② Třetím zařízením připojeným na systémovou sběrnici jsou čipy Intel 82441FX „PCI and Memory Controller“ (PMC) a Intel 82442FX „Data Bus Accelerator“ (DBX), které dohromady tvoří northbridge chipsetu Intel 440FX. Northbridge je připojen na všechny datové vodiče systémové sběrnice, ale jen na adresové vodiče HA3 až HA31 (počítáno od 0). Northbridge v sobě obsahuje řadič paměti pro SIMM paměťové moduly DRAM – paměťová sběrnice je paralelní s 64-bit datovou a dedikovanou 30-bit adresovou částí). Na základní desce je v SIMM modulech nainstalován plný 1 GB maximální northbridgem podporované kapacity DRAM. Součástí northbridge je též PCI host bridge pro paralelní 33 MHz 32-bit sběrnici PCI (sdílená 32-bit adresová a datová sběrnice). Sběrnice PCI má oddělený paměťový a I/O adresový prostor.

③ Na sběrnici PCI je připojen southbridge Intel 82371SB „PCI I/O IDE Xcelerator“ (PIIX3). Stejná PCI sběrnice je též vyvedena do 4 PCI slotů na základní desce. PIIX3 v sobě mimo jiné obsahuje PCI-ISA bridge pro 16-bit ISA sběrnici (paralelní 8 MHz sběrnice, 16-bit datová a dedikovaná 24-bit adresová sběrnice, zvláštní 16-bit I/O adresový prostor). Tato ISA sběrnice je vyvedena do 2 ISA slotů na základní desce.

④ Ve 2. PCI slotu je vložena grafická karta S3 Trio V64. Ve 3. PCI slotu je vložen USB 1.1 řadič (HBA) implementující OHCI – vystupující USB sběrnici označme jako B. Na sběrnici PCI je uvnitř PIIX3 southbridge připojen další (zabudovaný) USB 1.1 řadič (HBA) implementující UHCI (poznámka: UHCI není nekompatibilní s OHCI) – vystupující USB sběrnici označme jako A. Dále je uvnitř PIIX3 na sběrnici PCI připojen (zabudovaný) řadič jednoho kanálu sběrnice Parallel ATA.

⑤ Do kořenového hubu sběrnice USB A je připojen externí 1 GB flashdisk („fleška“) HDA (používá protokol Mass Storage) s jednou partition s ext2 FS a obsahuje na sobě instalaci Linuxu. Do kořenového hubu USB B je připojena USB 1.1 klávesnice. Na jediné sběrnici Parallel ATA je připojen 4 GB harddisk HDB jako master – disk obsahuje 2 partition: 1. partition HDB1 velká 1 GB je naformátována souborovým systémem NTFS, 2. partition HDB2 je velká 3 GB a je naformátována na ext2 FS.

Otázka č. 1 (X)

Vysvětlete, jaké jsou pravděpodobné důvody k tomu, že je uvedený northbridge připojený jen k adresovým vodičům 3 až 31 na systémové sběrnici.

Otázka č. 2 (X)

Pro dané PC bychom chtěli vyrobit **ISA** kartu s N GB přídavné paměti DRAM tak, aby tato paměť byla pro aplikace přístupná podobně jako 1 GB hlavní paměti RAM (a její použití bylo pro aplikace maximálně transparentní). Pro obě následující varianty rozhodněte a vysvětlete, zda by to bylo možné, a popište, jak by v takové variantě aplikace přistoupila např. k prvnímu a jak např. k poslednímu bajtu přídavné paměti RAM:

a) N = 1 GB
b) N = 4 GB

Otázka č. 3 (X)

Předpokládejte, že implementujeme jádro nového OS, který má běžet na výše uvedeném PC, a má podporovat práci s veškerými instalovanými zařízeními. Navrhněte (nakreslete obrázek) jakým způsobem bychom v takové situaci jádro OS strukturovali. Zaměřte se na podporu aplikačního souborového API, a aplikačního API pro čtení znaků z klávesnice minimálně ve výše uvedené situaci. Pokud budete kód jádra OS rozdělovat do více modulů, tak každý takový modul označte a stručně popište jeho funkci, a vyznačte všechny ostatní moduly a části OS, se kterými bude každý takový modul komunikovat při běžném požadavku od aplikace (např. čtení nebo zápis X bytů dat z/do nějakého souboru, čtení N znaků z klávesnice). Můžete předpokládat, že náš OS poběží jen na procesorové architektuře x86. Na druhou stranu ale předpokládejte, že bychom chtěli být schopni do OS maximálně jednoduše přidávat podporu pro další zařízení, a souborové systémy.

Otázka č. 4 (X)

Po zapnutí počítače v uvedeném nastavení a konfiguraci dojde k nabootování operačního systému Linux z připojeného externího flashdisku („flešky“). Popište, co se v počítači děje od jeho zapnutí do začátku provádění instrukcí jádra Linuxu. Popište hlavně to, jaké instrukce (a kterých programů) CPU po celou dobu startu počítače zpracovává, a odkud a jakým způsobem je přečte (případně jak se instrukce na dané místo dostanou).

Otázka č. 5

Napište program v jazyce Pascal (případně v jazyce C), který na standardní výstup (tj. pomocí procedury `WriteLn`) vypíše text „`Little Endian`“ nebo „`Big Endian`“ bez uvozovek podle toho, na jaké platformě bude spuštěn (resp. pro kterou bude přeložen). Připomenutí: prefixový unární operátor @ slouží v Pascalu pro získání adresy libovolné proměnné.

Otázka č. 6

Srovnejte termíny *cluster* a *cloud* a popište jejich výhody a nevýhody.

Otázka č. 7

Předpokládejte, že máme v Pascalu funkci s následujícím prototypem:

```
function NaFixedPoint(
      hodnota : integer; exp : integer
) : word;
```

Parametry funkce reprezentují reálné číslo x následujícím způsobem: `x = hodnota * 10`^{exp}. Funkce vrací hodnotu x jako reálné číslo s pevnou desetinou čárkou ve formátu 6+10. V šestnáctkové soustavě zapište návratovou hodnotu funkce `NaFixedPoint` po zavolání s následujícími parametry (reprezentujícími hodnotu `3,875`):

```
… := NaFixedPoint(3875, -3);
```

Předpokládejte, že typ `integer` je běžná Pascalová reprezentace celého 16-bitového znaménkového (signed) čísla, typ `word` je běžná Pascalová reprezentace celého 16-bitového bezznaménkového (unsigned) čísla.

Otázka č. 8

Předpokládejte, že v Pascalu implementujete část operačního systému s podporou pro vícevláknové zpracování a s preemptivním přepínáním vláken. Předpokládejte, že OS podporuje víceprocesorové SMP systémy, a vždy poběží na počítači s minimálně dvěma procesory (nebo procesorovými jádry). Vaším úkolem je naimplementovat podporu pro speciální druh zámků, tzv. *spinlock*, jehož využití je na daném systému někdy výhodné. *Spinlock* je běžný zámek ale s aktivním čekáním. Vaše implementace nemusí podporovat rekurzivní zamykání. Vaším úkolem je tedy v Pascalu naimplementovat následující záznam a k němu náležející procedury.
K dispozici máte fci CAS, která je standardní implementací operace *Compare & Swap* (resp. *Compare & Exchange*). Popište její chování a vhodně ji využijte ve vaší implementaci.

```
type
    TSpinLock = record
        ...
    end;
    PSpinLock = ^TSpinLock;

procedure Lock(lock : PSpinLock);
procedure Unlock(lock : PSpinLock);
```

Otázka č. 9

Předpokládejme, že implementujeme operační systém poskytující podporu pro vícevláknové zpracování, dynamicky linkované knihovny a stránkování (tj. běží jen na procesorových platformách s podporou stránkování). Předpokládejme situaci, že více procesů běžících v takovém systému často používá stejnou dynamicky linkovanou knihovnu. Abychom ušetřili paměť, tak bychom chtěli, aby v takovém případě byl minimálně kód knihovny uložený v paměti RAM jen jednou. Je možné to nějak zařídit? Pokud ano, popište přesně, jakým způsobem by to bylo možné, a jak bude v takovém systému probíhat spouštění aplikace používající takovou „sdílenou“ DLL. Pokud ne, popište přesně všechny důvody, proč není možné takový systém naimplementovat.

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 `0x4000` v paměťovém 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`. Proměnná `b` se skládá z následujících částí: bity 2 až 7 reprezentují znaménkové 6-bitové celé číslo C **ve dvojkovém doplňku**, bit 0 je příznakem pojmenovaným `F0`, bit 1 je příznakem `F1`.
Účelem této procedury je **najednou (v jednom okamžiku)** změnit stav GPIO signálů „vysílaných“ GPIO řadičem následujícím způsobem (vše počítáno od `0`):

- 0. až 5. GPIO na absolutní hodnotu čísla C
- 29. GPIO na `0`, když je `F0` rovno `1`; a na `1`, když je `F0` rovno `0`
- 30. GPIO na hodnotu `F1`
- 31. GPIO na `0`
- Ostatní GPIO linky musí zůstat nezměněné!

Váš kód vždy poběží na počítači s procesorem Intel Pentium III (32-bitový procesor, 64-bit datová sběrnice, 36-bit adresová sběrnice, separátní 16-bit I/O adresový prostor). Pořadí bitů i bytů chápané procesorem, sběrnicí PCI, i řadičem GPIO jsou stejná. Předpokládejte, že stránkování, ani segmentace nejsou zapnuté.

Pozor: instrukce `MOV` má u procesoru Intel Pentium III kromě varianty s **32-bit** operandy i varianty s **8-bit, resp. 16-bit** operandy, které zde ale **nesmíme** použít, přestože na první pohled vypadá jejich použití jako ekvivalentní – protože např. čtyři postupné 8-bitové zápisy na adresy `0x4000`, `0x4001`, `0x4002`, `0x4003` nejsou pro nás ekvivalentní jednomu 32-bitovému zápisu na adresu `0x4000`, protože **způsobí jen postupnou změnu** potřebných linek GPIO. Máte ale zaručeno, že použitý překladač Pascalu vždy generuje instrukci `MOV` s 32-bitovým operandem, pokud do paměti zapisujete (nebo čtete) zarovnanou 32-bitovou hodnotu!