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.