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. Do zápatí všech listů vždy napište i/N (kde i je číslo listu, N je celkový počet odevzdaných listů).

Otázka č. 1

Typický firmware moderních počítačů PC bude jistě obsahovat nějakou API funkci pro zjištění struktury fyzického adresového prostoru, nazvěme ji např. `GetMemoryMap`. Pokud by taková funkce/procedura byla naprogramována v Pascalu, napište, jak by mohla vypadat její deklarace a detailně rozmyslete, jak bude vypadat datová struktura s popisem adresového prostoru, kterou vyplňuje – dbejte na rozumnou paměťovou úspornost datové struktury, aby mohla být reálně použitelná. Navrhněte nějakou realistickou situaci a pro ni uveďte příklad konkrétních dat vyplněných ve struktuře vrácené funkcí `GetMemoryMap`. Předpokládejte, že cílový počítač používá procesor Intel Pentium (s architekturou IA-32) a jako systémová sběrnice v něm slouží nejběžnější varianta sběrnice PCI.

Společná část pro otázky označené X

Předpokládejte, následující program v Pascalu uložený ve zdrojovém souboru `App.pas`:

```
type
  PItem = ^TItem;
  TItem = record
    Data : array[0..511] of longint;
    Next : PItem;
  end;

var
  list : PItem;

function AddNewItem(var list : PItem) : PItem;
var item : PItem;
begin
  New(item);
  item^.Next := list;
  list := item;
  AddNewItem := item;
end;

begin
  list := nil;
  AddNewItem(list);
  AddNewItem(list);
  AddNewItem(list);
  AddNewItem(list);
  AddNewItem(list);
  ...
end.
```

Otázka č. 2 (X)

Předpokládejte, že uvedený program přeložíme ve Free Pascalu (`longint` je 32-bitový celočíselný znaménkový typ), a spustíme ho na 32-bitových Windows 7 na počítači s procesorem Intel i7 a se 3 GB operační paměti. V průběhu programu se provede 5 dynamických alokací proměnné `item` – detailně vysvětlete, jak asi procedura `New` funguje, kde se „sebere“ vrácenou „dynamickou“ paměť. Pokud je to nějaký složitější proces, tak vysvětlete všechny úrovně až k místu rozhodnutí, na jakých **fyzických** adresách budou data dynamicky alokovaných proměnných ležet.

Otázka č. 3 (X)

Pro situaci z otázky 2 detailně vysvětlete, jak a v jakých fázích bude probíhat překlad programu App.pas, a jaký všechen kód se bude překladu účastnit, a jaký kód bude součástí výsledného spustitelného souboru. Pokud máte někde na výběr více variant, tak vyberte tu nejpravděpodobnější, jak by se asi překladač (a případné navazující nástroje) Free Pascalu mohl chovat.

Otázka č. 4 (X)

Detailně vysvětlete, zda je uvedená datová struktura jednosměrně vázaného seznamu a funkce `AddNewItem` thread-safe. Pokud ano, tak vysvětlete proč. Pokud ne, tak napište příklad race-condition, která při provádění ve více vláknech může vzniknout.

Otázka č. 5 (X)

Předpokládejte, že máme počítač s variantou 32-bitového procesoru Intel 80386 běžícího v 32-bitovém režimu s vypnutou podporou pro stránkování (virtuální i fyzický adresový prostor procesoru má šířku 32-bitů, virtuální/logická adresa se přímo rovná adrese fyzické). Procesor má obecnou registrovou architekturu, a mimo jiné 7 obecných 32-bitových registrů `EAX`, `EBX`, `ECX`, `EDX`, `ESI`, `EDI`, `EBP`, dále má 32-bitový registr `ESP` (stack pointer), 32-bitový registr `EIP` (program counter), a příznakový registr s běžnými příznaky. ISA obsahuje instrukce `MOV` (má 3 varianty: load, store, přesun mezi dvěma registry), `PUSH`, `POP`, `ADD` (sčítání bez přenosu), `SUB` (odečítání bez přenosu), `CALL`, a `RET` s běžnou sémantikou. Instrukce `MOV`, `ADD`, `SUB` mají dva argumenty (maximálně jeden smí být typu adresa – viz níže), `PUSH`, `POP`, `CALL` mají jeden argument, instrukce `RET` je bez argumentů. Argumentem uvedených instrukcí může být immediate (kde to dává smysl), libovolný obecný registr nebo registr `ESP`, nebo adresa (formát viz níže). Pro tento počítač jsme přeložili výše uvedený program do strojového kódu, přičemž víme, že náš překladač Pascalu používá variantu Cčkové volací konvence, kdy jsou argumenty funkce ukládány na zásobník (odstraňuje je volající), návratová hodnota je v registru `EAX`, volaný musí volajícímu zachovat minimálně obsah registru `EBP`. Pokud bychom nyní ze strojového kódu disassemblovali funkci `AddNewItem`, napište v Intel syntaxi assembleru, jak by mohl vypadat její překladačem vygenerovaný kód. Intel syntaxe pro uvedený procesor je následující: cílový operand instrukce je vždy nejvíce vlevo; immediate operand je číslo bez prefixů v desítkové nebo šestnáctkové soustavě (přípona `h`), hodnota v hranatých závorkách je operand typu adresa, tj. např. `[imm/reg]` znamená hodnotu operandu na adrese `imm/reg`, což je nejjednodušší varianta nejobecnějšího procesorem podporovaného operandu typu adresa: `[reg1 + (reg2 * imm2) + imm1]`, který znamená hodnotu na adrese spočítané jako výsledek výrazu v hranatých závorkách. Platí: `imm` jsou immediate, `imm2` může být pouze `1`, `2`, `4`, nebo `8`, `reg` může být obecný registr nebo registr `ESP`. Všechny části výrazu jsou nepovinné.

Otázka č. 6

Detailně vysvětlete, proč se často pro některé signální vodiče používá inverzní logika, tj. že `0` znamená `TRUE`, a `1` znamená `FALSE`. Lze nějak takový signál poznat jen z jeho označení? Proč se inverzní logika používá jen pro některé vodiče nějaké sběrnice a nikoliv pro všechny?

Otázka č. 7

Předpokládejte, že programujeme aplikaci pro speleology, jejímž úkolem bude z obrázku stropu krasové jeskyně rozpoznat počet visících jedinců ohroženého netopýra vrápence malého. Naše aplikace bude jednokanálový obrázek stropu jeskyně zachycený infračervenou kamerou (hodnota každého pixelu je intenzita zachyceného světla v infračerveném spektru) načítat do níže uvedené globální proměnné `buffer`. Pokud z infrakamery získáváme pouze čtvercové obrázky, tak rozhodněte a zdůvodněte, v jakém největším rozlišení se nám obrázek do proměnné `buffer` vejde. Lišil by se nějak maximální rozměr obrázku, pokud bychom jeskyni snímali barevnou denní kamerou s přisvícením halogenovým reflektorem? Pokud ano, tak jak. Pokud byste potřebovali znát nějaké další charakteristiky zaznamenaného obrazu, tak si je vhodně zvolte, a uveďte je.

```
var
  buffer : array[0..65535] of byte;
```

Společná část pro otázky označené Y

Zakládáme 1. matfyzácký řetězec s rychlým občerstvením *MFC* (*Malostranský Fried Cheese*), kde kromě smažáku bude důležitým prodejním artiklem i bohatá nabídka nápojů od společnosti *Pepsi Co*. Pro prodej nápojů jsme se rozhodli převzít model našeho nejbližšího konkurenta (*KFC*), kde si zákazníci mohou do kelímku natočit libovolné množství vybraného nápoje. Pro naše restaurace tedy vyrábíme podobný postmix s dotekovou obrazovkou (viz obrázek vlevo dole) s rozlišením `576x1024` pixelů, kde bude po zapnutí neustále zobrazena nabídka nápojů (každému odpovídá jedna čtvercová výběrní plocha `180x180` pixelů) – tekutina má z postmixu téct právě tehdy, když se uživatel dotýká prstem obrazovky v místě obrázku nápoje, **pokud zároveň není vyčerpán sirup dané příchutě**. Do výstupní trysky postmixu je zaveden vstup kohoutkové vody (ovládaný elektromagnetickým ventilem `Vvoda`), vstup vody obohacené o oxid uhličitý tedy CO_2 (ovládaný ventilem `Vsoda`), a 5 trysek ze zásobníků se sirupy jednotlivých příchutí (ventily `V1` až `V5`). Každý z ventilů je spolehlivě otevřen po přivedení napětí minimálně `3V`, a je spolehlivě uzavřen při napětí maximálně `1V`. Nabídka nápojů bude následující (v závorce uvedena souřadnice levého horního rohu ovládacího čtverce, a číslo ovládacího ventilu): *Pepsi* (`198,60→1`), *Pepsi Light* (`85,280→2`), *7UP* (`310,280→3`), *Mirinda* (`85,490→4`), *Lipton Ice Tea* (`310,490→5`) – pouze ledový čaj se mixuje s čistou vodou, vše ostatní se mixuje se sodou. Navíc je možnost natočit si čistou vodu (`85,700`) a čistou sodu (`310,700`). V každém z 5 zásobníků na sirup je instalován senzor skoro nulové hladiny sirupu (`S1` až `S5`), který na svůj výstupní signál připojuje zem právě tehdy, když je zásoba sirupu vyčerpána (jinak je výstupní signál odpojen).

Otázka č. 8 (Y)

Pro řízení postmixu používáme 32-bitový µC se zabudovaným řadičem I^2C sběrnice, a s řadičem 32 digitálních vstupně výstupních GPIO pinů procesoru (`IO0` až `IO31`). Ke každému GPIO pinu je připojený vhodný pull-up odpor na napětí `5V`. Řadič GPIO má jeden 32-bit vstupní read-only registr paměťově mapovaný na adrese `0xFF000000`, hodnota v n-tém bitu odpovídá aktuálně naměřené hodnotě na pinu `IOn` (více něž `2V` → `1`, méně než `1V` → `0`). Dále má řadič na adrese `0xFF000000` 32-bit výstupní write-only registr, kde n-tý bit řídí stav pinu `IOn` (`1` = v µC je pin `IOn` odpojen, `0` = pin připojen na zem). Na pinu `IO0` je připojen ventil `Vvoda`, piny `IO1` až `IO5` → ventily `V1` až `V5`, pin `IO6` → ventil `Vsoda`, piny `IO9` až `IO13` ← senzory `S1` až `S5`. Na I^2C sběrnici je připojen řadič dotekové vrstvy obrazovky. V Pascalu již máme naprogramovánu funkci:

```
function GetTouchInfo(
  var x : longword; var y : longword) : boolean;
```

která pomocí I^2C řadiče přečte poslední zaznamenanou událost na dotekové vrstvě (`x` a `y` jsou souřadnice události, návratová hodnota: `true` = zaznamenán dotek, `false` = zaznamenána ztráta doteku). Naprogramujte v Pascalu s jejím použitím firmware pro uvedený počítač tak, aby se choval přesně podle specifikace postmixu uvedené výše.

Otázka č. 9 (Y)

Předpokládejte, že bychom na výše uvedeném procesoru, chtěli nabootovat variantu OS Linux, a chování postmixu naprogramovat jako běžnou Linux aplikaci. Zároveň předpokládejte, že jádro tohoto Linuxu kromě běžných API funkcí nemá API funkce pro práci s GPIO ani I^2C řadiči. Pokud bychom chtěli mít v jádře Linuxu co nejmenší počet různých API funkcí, tak pro něj navrhněte a detailně popište, jak by se s ovladači GPIO a I^2C mělo z běžné aplikace komunikovat.

Otázka č. 10

Spustitelný EXE soubor OS MS-DOS začíná hlavičkou:

Offset	|B|	Popis
`0x00`	2	ASCII řetězec „MZ“
`0x02`	2	Počet bytů v posledním 512 B bloku
`0x04`	2	Délka souboru v celých 512 B blocích

OS Windows používá formát spustitelných souborů PE (*Portable Executable*) – každý takový soubor vypadá tak, že začíná daty standardního MS-DOS EXE souboru, ale po délce indikované v jeho hlavičce soubor nekončí, ale následuje PE hlavička (jejíž první 4 byty jsou `$50,$45,$00,$00`). Oba formáty jsou *little endian*. Napište v Pascalu program (fungující na LE i BE strojích), který jako svůj 1. argument – dostupný jako výsledek běžné Pascal funkce `ParamStr(1)` – dostane jméno binárního souboru. Program má poznat druh souboru a vypsat jeden z řetězců: `DOS`, `PE`, nebo `unknown`.