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 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. 2

Předpokládejte N-bitový procesor s X-bitovým paměťovým adresovým prostorem (kde, šířka vnější adresové sběrnice je X bitů, a šířka vnější datové sběrnice je Y bitů). Např. u mikroprocesoru Intel 8088 je N=16, X=20, Y=8. V tomto kontextu odpovězte na následující:

a) Nejprve obecně a pak na uvedeném příkladu vysvětlete, co je slovo (word) a jaká je jeho velikost?
b) Velmi často se můžete setkat s tím, že jako velikost jednoho slova se chápe konkrétní konstanta, která nemusí odpovídat obecné definici slova podle bodu a). Napište, jaká tato konstanta je, a vysvětlete, proč se používá.

Otázka č. 3

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

Otázka č. 4

Předpokládejte nějaký typický formát spustitelného souboru pro nějaký běžný operační systém. Z jakých částí se takový spustitelný soubor skládá? Pro každou takovou část popište její obsah a funkci.

Otázka č. 5

Popište, co vše zahrnuje a určuje volací konvence procedur a funkcí. Uveďte příklad jedné běžně používané volací konvence a detailně ji popište.

Otázka č. 6

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 25):

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

vlákno *T2* (priorita 31):

```
lock A
lock C
lock B
opShort
unlock B
unlock C
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. 7
(otázka za celkem 2 body)

Předpokládejte počítač s 32-bitovým paměťovým adresovým prostorem. V systému je nainstalována 1 MB velká paměť ROM, která je souvisle namapována na nejvyšší možné adresy v paměťovém adresovém prostoru počítače. Dále je v systému nainstalováno 512 MB paměti RAM, která je souvisle namapována od adresy `0` v paměťovém adresovém prostoru počítače, s výjimkou adres `A000h` až `A007h` na kterých jsou namapovány porty řadiče pevných disků 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 (24-bitové číslo uložené ve spodních 24 bitech parametru `sector`), 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 : Longword;
               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) `A001h`: horních 8 bitů čísla sektoru
2) `A002h`: prostředních 8 bitů čísla sektoru
3) `A003h`: spodních 8 bitů čísla sektoru

Poté je třeba vyčkat, až bude řadič připraven na přijímání dat – to řadič indikuje ve stavovém portu na adrese `A000h` nastavením jeho 6. bitu (číslované od 0), tzv. BSY (Busy), na hodnotu `0` současně s nastavením 0. bitu, tzv. DRQ (Data Request), na hodnotu `1`. Poté je možné zapisovat jednotlivé byty (které mají být zapsány do vybraného sektoru) do datového portu na adrese `A007h`. Zápis každého bytu do datového portu způsobí nastavení bitů BSY na hodnotu `1` a DRQ na hodnotu `0`. Před zápisem každého dalšího datového bytu je tedy třeba vždy vyčkat, až řadič opět oznámí svoji připravenost současným nastavením BSY na `0` a DRQ na `1`.

Každé současné nastavení bitů BSY na `0` a DRQ na `1` je řadičem disku indikováno vyvoláním přerušení číslo `14`. V obsluze přerušení `14` je třeba ověřit, že přerušení opravdu vyvolal řadič disků a ne jiný zdroj, tj. že hodnota bitů BSY a DRQ je opravdu `0` a `1`. Pokud přerušení pochází z jiného zdroje, je možné ho ignorovat.

Předpokládejte, že není zapnutá segmentace, ani stránkování. Předpokládejte, že typ `Longword` slouží pro ukládání bezznaménkových celých čísel a jeho velikost je 32 bitů.

Otázka č. 8

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. 9

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.