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
(otázka za celkem 2 body)

Předpokládejte počítač s 32-bitovým paměťovým adresovým prostorem. V systému je nainstalována 256 kB velká paměť ROM, která je souvisle namapována na nejvyšší možné adresy v paměťovém adresovém prostoru počítače. Dále je v systému nainstalován 1 GB paměti RAM, která je souvisle namapována od adresy `0` v paměťovém adresovém prostoru počítače, s výjimkou adres `1F0h` až `1F7h` 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 procedury s následujícím prototypem (viz níže), která má zapsat 1 sektor 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):

```
type
      PByte = ^Byte;
procedure Write(sector : Word;
                data : PByte);
```

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) `1F3h`: spodních 8 bitů čísla sektoru
2) `1F4h`: horních 8 bitů čísla sektoru
3) `1F7h`: 8-bitový identifikátor příkazu: příkaz `WRITE SECTOR` = hodnota `30h`

Poté je třeba vyčkat, až bude řadič připraven na přijímání dat – to řadič indikuje nastavením 3. bitu (číslované od 0), tzv. DRQ (Data Request), ve stavovém portu na adrese `1F7h` na hodnotu `1`. Poté je možné zapisovat jednotlivé byty (které mají být zapsány do vybraného sektoru) do datového portu na adrese `1F0h`. Zápis každého bytu do datového portu způsobí nastavení bitu DRQ na hodnotu `0`. Před zápisem každého dalšího datového bytu je tedy třeba vždy vyčkat, až řadič opět oznámí svoji připravenost nastavením DRQ na `1`.

Řadič negeneruje žádná přerušení, a pro komunikaci s ním je tedy třeba použít mechanizmus pollingu. Předpokládejte, že není zapnutá segmentace, ani stránkování. Předpokládejte, že typ `Word` slouží pro ukládání bezznaménkových celých čísel a jeho velikost je 16-bitů.

Otázka č. 2

Procesor 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) označený písmenem C.

Procesor 6502 má následující instrukce:

- LDA *$adresa/#konstanta* (Load Accumulator) – načtení jednobytové hodnoty na zadané adrese (argument instrukce uvozený $) *nebo* přímo zadané jednobytové konstanty (argument instrukce uvozený #) do akumulátoru.
- STA *$adresa* (Store Accumulator) – uložení hodnoty akumulátoru na zadanou 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 (jednobytové proměnné A, B, C v sobě obsahují celá bezznaménková čísla a jsou uloženy na následujících adresách: A = `0A00h`, B = `0B00h`, C = `0C00h`). Všechny operace chceme provést v celých číslech s 8-bitovou přesností bez znaménka:

```
A := B + B + C - 16
```

Otázka č. 3

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 Big 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. 4

Následující hodnotu:

`3,375`

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.

Otázka č. 5

Předpokládejme, že v operačním systému poskytujícím podporu pro vícevláknové zpracování (s preemptivním přepínáním vláken) chceme naimplementovat podporu pro UNIXové signály, konkrétně proceduru `Signal(id : ThreadId; signalNum : Integer)`, která způsobí, že cílové vlákno identifikované parametrem `id` obdrží signál s číslem `signalNum`. Předpokládejme, že cílové vlákno si pro dané číslo signálu již u operačního systému zaregistrovalo svoji obsluhu (adresu obslužné procedury, která se v něm má zavolat). Popište, jak bude procedura `Signal` naimplementovaná (tj. jakým způsobem přesně zařídí, že v cílovém vlákně dojde k vyvolání obslužné procedury). Zvlášť rozeberte chování v případě, že je cílové vlákno ve stavu „ready-to-run“, a zvlášť případ, že je ve stavu „running“. Předpokládejte, že cílové vlákno je vždy jiné než vlákno volající proceduru `Signal`.

Otázka č. 6

Předpokládejme, že máme překladač jazyka Pascal pro procesorovou architekturu a OS, které podporují mechanizmus segmentace. Dále předpokládejme následující část programu v Pascalu, který budeme pro takový cílový systém překládat a výslednou aplikaci na něm spouštět:

```
type
      PByte = ^Byte;
var
      uk : PByte;
begin
      New(uk);
      uk^ := 10;
      …
end.
```

Vyberte si nějakou vhodnou velikost adresových prostorů a v tomto kontextu popište, jaké informace bude obsahovat proměnná `uk`, a kolik bytů bude tato proměnná zabírat v paměti (pozor, zajímá nás velikost samotné proměnné `uk`, nikoliv velikost dat na které ukazuje). Dále na příkladu konkrétních hodnot vysvětlete, jaké všechny operace se stanou při vykonávání kódu řádku „`uk^ := 10;`“ a kdo je bude provádět (tj. kdo a jakým způsobem určí adresu, na kterou se zapíše uvedená hodnota `10`, a kterou uvidí řadič paměti).

Otázka č. 7

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 každé takové aplikace?

Otázka č. 8

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. 9

Popište a vysvětlete termíny prolog a epilog procedury/funkce. Co přesně je jejich funkcí a kdo zařídí jejich vytvoření?