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 počítač s 32-bitovým paměťovým adresovým prostorem. Architektura počítače zahrnuje zvláštní adresový prostor pro komunikaci mezi procesorem a připojenými zařízeními. 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 nainstalováno 1 GB paměti RAM, která je souvisle namapována od adresy `0` v paměťovém adresovém prostoru počítače, s výjimkou adres `B8000h` až `B8F9Fh` na kterých je namapována textová video paměť grafické karty 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 vypíše textový řetězec na určené místo na obrazovce (hodnota `x` je `0` – 79, hodnota `y` je `0` – 24). Organizace video paměti je následující: grafická karta podporuje pouze jeden textový režim s rozlišením 80 * 25 znaků (tj. 80 znaků na řádek, 25 řádků na obrazovku), pro každý znak na obrazovce jsou ve video paměti vyhrazeny 2 byty (v 1. bytu je kód znaku, který se má zobrazit, 2. byte je rezervovaný a vždy by měl mít hodnotu 8h), informace o jednotlivých znacích na obrazovce jsou ve video paměti uloženy souvisle zleva doprava a shora dolů (tj. na adrese `B8000h` je informace pro 0. znak zleva na 0. řádku). Můžete předpokládat, že text se vždy celý vejde na zadaný řádek od zadané pozice. 8-bitové kódování znaků použité pro řetězec `text` se shoduje s kódováním, které používá grafická karta. Předpokládejte, že v průběhu celé vaší funkce jsou zakázána všechna přerušení a NMI nemůže vzniknout. Předpokládejte, že není zapnutá segmentace, ani stránkování. Předpokládejte, že typ `Longword` slouží pro ukládání bezznaménkových celých čísel a jeho velikost je 32-bitů.

```
procedure Print(x : Longword;
                y : Longword;
                text : String);
```

Otázka č. 2

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

Otázka č. 3

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

```
(3 SHL 2) XOR (65536 OR 9)
```

Otázka č. 4

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 = `$8000`, B = `$1000`, C = `$4000`):

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

Otázka č. 5

V kódování Unicode existuje následující přiřazení kódů jednotlivým znakům: kód `48h` pro znak ‚H‘, kód `61h` pro znak ‚a‘, kód `6Ch` pro znak ‚l‘, kód `6Fh` pro znak ‚o‘, kód `F3h` pro znak ‚ó‘, a kód `301h` pro kombinující znak (combining character) diakritické znaménko čárka (stejné znaménko jaké je použito u znaku ‚ó‘).

a) Je v kódování Unicode nějaký významový rozdíl mezi znakem `F3h` a posloupností znaků `6Fh 301h`? Pokud ano, tak jaký?

b) Od adresy `0` chceme do paměti uložit text „`Haló`“ (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. 6

Předpokládejte provedení následující instrukce:

```
MOVE.W      [2047], D1
```

která načítá 2 bytovou hodnotu z adresy `2047` do registru `D1`. Operační kód instrukce začíná na adrese `4090` a její celková délka je 7 bytů. Systém používá stránky o velikosti 2 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.

Otázka č. 7

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, a jak a kdy přesně dojde k vyvolání obslužné procedury signálu v cílovém vlákně. 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. 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`):

```
ř22: 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
ř23: b := 0;
      C7 05 30 C0 40 00 00 00 00 00
            mov   [0040C030h], 0
ř24: 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 23 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 [0040C030h], 0`?
b) Jak debugger způsobí, že se program „zastaví“ před provedením instrukce `mov [0040C030h], 0`, když přeci procesor stále musí nějaký kód vykonávat?

Otázka č. 9

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 Windows 7 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. 10

Popište, jaké všechny podmínky musí být splněny, aby mohlo dojít ve vícevláknové aplikaci k deadlocku.