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 a v systému se nepoužívá žádné memory-mapped I/O. 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 N bytů paměti RAM, která je souvisle namapována od adresy 0 v paměťovém adresovém prostoru počítače. Máte jistotu, že součet velikostí nainstalované paměti RAM a paměti ROM nepřesahuje velikost paměťového adresového prostoru. Předpokládejte, že v Pascalu implementujete část firmware počítače, který bude uložený ve zmíněné paměti ROM. Napište implementaci funkce s následujícím prototypem (viz níže), která vrací velikost instalované paměti RAM (tj. vrací číslo N). Předpokládejte, že překladač Pascalu generuje pro vaši funkci takový strojový kód, že pro všechny lokální proměnné používá pouze registry procesoru a nepotřebuje použít žádnou část paměti RAM. 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.

```
function GetAvailableMemory : Longword;
```

Otázka č. 2

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 = `$1000`, B = `$1004`, C = `$2000`):

```
A := (B + 3) * (C + 1)
```

Otázka č. 3

Popište, co znamená pojem PIO, a vysvětlete, jak daný koncept funguje.

Otázka č. 4

Spočítejte hodnotu následujícího výrazu zapsaného v Pascalu (předpokládejte, že celý výpočet i všechny uvedené hodnoty jsou v 16-bitových celých číslech bez znaménka):

```
1024 OR (7 AND NOT(3))
```

Otázka č. 5

Předpokládejte provedení následující instrukce:

```
MOV   EAX, [4094]
```

která načítá 4 bytovou hodnotu z adresy 4094 do registru EAX. Operační kód instrukce začíná na adrese 314 a její celková délka je 6 bytů. Systém používá stránky o velikosti 1 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. 6

Popište problém priority inversion, a navrhněte způsob, jak se mu bránit.

Otázka č. 7

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 7 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 pro daný scénář bude potřeba více různých spustitelných souborů daného programu, nebo zda bude dostačovat pouze jeden.

```
program Faktorial;
var
  f, n : integer;

begin
  ReadLn(n);
  f := 1;
  while n > 1 do begin
    f := f * n;
    Dec(n);
  end;
  WriteLn('Faktorial je ', f);
end.
```

Otázka č. 8

Předpokládejme, že v operačním systému poskytujícím podporu pro vícevláknové zpracování chceme naimplementovat proceduru `Sleep(ms : Longint)`, která způsobí, že vlákno, které ji zavolá, nebude ve zpracování dalších instrukcí pokračovat dříve než za `ms` milisekund. Popište 2 základní varianty možné implementace dané procedury a rozeberte jejich výhody a nevýhody.

Otázka č. 9

Předpokládejte programové prostředí, které poskytuje podporu pro vícevláknové zpracování. Předpokládejte, že hlavní procedura nějakého vlákna doběhne do konce (tj. toto vlákno je ve stavu těsně před provedením instrukce RET [instrukce návratu z podprogramu] na konci jeho hlavní procedury). Popište a vysvětlete, co se bude dít potom.

Otázka č. 10

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?