

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 nějaký 32-bitový procesor s load-store architekturou, s obecnými registry R0 až R7, a běžnými speciálními registry. Tento procesor má v instrukční sadě instrukci `cmp` (compare) a běžné instrukce podmíněného skoku. Popište a vysvětlete, jaké argumenty bude instrukce `cmp` mít, a jaké bude chování instrukce `cmp` ve všech typických situacích.

Otázka č. 2

Následující procedura `MapAll` naprogramovaná v jazyce Pascal zpracuje všechny položky pole předaného v parametru `data` tak, že na každou zavolá proceduru jejíž adresa jí byla předána v parametru `mapProc`:

type

```
TValueArray = array[0..N-1] of Pointer;
PValueArray = ^TValueArray;
PMapProc = procedure (value : Pointer);
```

procedure `MapAll`(

```
data : PValueArray; mapProc : PMapProc
```

);

var

```
i : Integer;
```

begin

```
for i := 0 to Length(data^) - 1 do begin
  mapProc(data^[i]);
```

```
end;
```

end;

Předpokládejte, že proceduru `MapAll` budeme používat pouze v programech implementovaných pro nějaký konkrétní OS s podporou vícevláknového zpracování, a že ji budeme často volat na víceprocesorovém (resp. vícejádrovém) systému. Napište novou lepší implementaci procedury `MapAll` (v Pascalu, případně v jazyce C) tak, aby na nevytíženém systému plně využívala výkonu všech dostupných procesorových jader. Předpokládejte, že na pořadí volání procedury `mapProc` na jednotlivých položkách pole `data` nezávisí, a že celá procedura `MapAll` smí skončit, až když byla volání procedury `mapProc` dokončena na všech položkách pole `data`.

Všechny potřebné API funkce/procedury OS pro podporu vícevláknového zpracování si vhodně navrhnete sami. Pro každou takovou funkci/proceduru OS jen stručně popište, co dělá, ale neimplementujte ji.

Otázka č. 3

Srovnejte analogový a digitální přenos dat, každý z nich popište, a uveďte jejich výhody a nevýhody.

Otázka č. 4

Detailně vysvětlete koncept DLL. Co znamená provádět jejich linkování, kdo a v jaké situaci ho provádí, jak proces linkování přesně probíhá, a jaké všechny informace jsou k němu potřeba?

Otázka č. 5

Předpokládejte, že implementujete funkci OS, která pošle N bytů dat po lokální síti. Funkce OS dostane od aplikace v jednom parametru ukazatel na první byte, který má po síti poslat, a ve druhém parametru číslo N. Pro komunikaci se síťovou kartou (pro zápis do jejích registrů) se používá mechanismus *port-mapped IO*, a síťová karta používá mechanismus DMA pro přenos dat z/do hlavní paměti RAM. Před začátkem zápisu tedy musí OS do registrů síťové karty zapsat zdrojovou adresu v paměti RAM, která ukazuje na data připravená pro odeslání. Kvůli optimalizaci výkonu bychom ale chtěli podporovat jen situaci, kdy síťová karta vždy získá přímo adresu původních dat ve zdrojové aplikaci (tj. před provedením operace odeslání nikdy nechceme data kopírovat na jiné místo v paměti). Za předpokladu, že se v OS používá mechanismus stránkování, vysvětlete následující:

- Vysvětlete, co vše musí OS v takové situaci udělat, aby síťové kartě předal správnou adresu. Na příkladu uveďte jakou.
- Bude tento princip fungovat pro všechny možné adresy a hodnoty N, které může aplikace operačnímu systému předat? Vysvětlete proč ano, resp. proč ne.

Otázka č. 6

Srovnejte termíny/technologie ROM, PROM, EPROM, EEPROM a Flash ROM.

Otázka č. 7

Následující obrázek obsahuje část screenshotu hex editoru, který zobrazuje obsah 59 bytů dlouhého binárního souboru:

	0001	0203	0405	0607	0809	0A0B	0C0D	0E0F
00	0000	0000	0000	F07F	0000	0000	0000	F0FF
10	0000	8044	50C5	99C3	AD6C	69C5	A120	C5BE
20	6C75	C5A5	6F75	C48D	6BC3	BD20	6BC5	AFC5
30	8800	0000	3E72	00E1	0064	00		

Víme, že všechna data jsou v souboru uložena jako little endian, a že od 16. bytu (počítáno od 0) je v souboru uloženo 32-bitové reálné číslo s pohyblivou desetinnou čárkou. Dále je od 49. bytu v souboru uloženo 2. reálné číslo ve stejném formátu. Mantisa je u obou 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 **obou** těchto reálných čísel v desítkové soustavě.

Otázka č. 8

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

$(\$1000 \text{ OR } 10) \text{ XOR } (3 \text{ SHL } \$4)$

Otázka č. 9

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):

```
ř15: 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
ř16: b := 0;
      C7 05 30 C0 40 00 00 00 00 00
          mov    [0040C030h], 0
ř17: 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 17 umístit breakpoint. V tomto kontextu odpovězte na následující otázky:

- 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 eax, [0040C020h]`?
- Jak debugger způsobí, že se program „zastaví“ před provedením instrukce `mov eax, [0040C020h]`, když přeci procesor stále musí nějaký kód vykonávat?

Otázka č. 10

Předpokládejte, že v OS s podporou pro vícevláknové zpracování dojde k náhlému ukončení nějakého vlákna (např. po dereferenci neplatného ukazatele) v situaci, kdy toto vlákno drží několik zamčených zámků. Implementace zámků je poskytována operačním systémem. Jak se v takové situaci má OS zachovat? Popište všechny typické možnosti řešení daného problému a vysvětlete jejich výhody a nevýhody.