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. Do zápatí všech listů vždy napište i/N (kde i je číslo listu, N je celkový počet odevzdaných listů).

Společná část pro otázky označené X

Předpokládejte, že vyrábíme embedded systém (počítač) s následující architekturou (systém obsahuje i další níže neuvedená zařízení):

① Hlavní systémová sběrnice je paralelní 66 MHz sběrnice s 36-bit adresovou a dedikovanou 64-bit datovou částí. Na systémovou sběrnici je připojen CPU Intel Pentium Pro (32-bit procesor s instrukční sadou x86, s 36-bit paměťovým adresovým prostorem, 16-bit I/O adresovým prostorem, 64-bit datovou sběrnicí; taktovací frekvence jádra je 166 MHz.

② Druhým zařízením připojeným na systémovou sběrnici jsou čipy Intel 82441FX „PCI and Memory Controller“ (PMC) a Intel 82442FX „Data Bus Accelerator“ (DBX), které dohromady tvoří northbridge chipsetu Intel 440FX. Northbridge v sobě obsahuje řadič paměti pro SIMM paměťové moduly DRAM – paměťová sběrnice je paralelní s 64-bit datovou a dedikovanou 30-bit adresovou částí). Součástí northbridge je též Host/PCI bridge pro standardní 33 MHz 32-bit sběrnici PCI.

③ Na sběrnici PCI je připojen kombinovaný USB 2.0 řadič NXP Semiconductors SAF1562HL – pro komunikaci s USB 1.1 zařízeními implementuje OHCI rozhraní, pro komunikaci s USB 2.0 zařízeními implementuje EHCI rozhraní. Na jednom z portů jeho kořenového hubu je připojen USB 1.1 touchpad (chovající se z pohledu SW jako USB myš). Na druhém portu kořenového hubu je připojen flash disk s jedním bootovatelným oddílem se souborovým systémem ext2fs, na kterém je nainstalován OS.

④ Dále je na sběrnici PCI připojen Calibre PCI93LV PCI/ I^2C HBA. Na I^2C sběrnici se používá 7-bitové adresování. Ke sběrnici I^2C je připojena single-touch dotyková vrstva připojeného displeje (touchscreen) na adrese $4F. Při komunikaci s touchscreen se používá standardních HID paketů, které se přímo posílají po I^2C sběrnici dle standardu HID Over I^2C (před standardní HID hlavičky paketu je jen přidána 2 bytová informace o délce paketu).

Otázka č. 1 (X)

Pro uvedenou PCI sběrnici nakreslete časový diagram průběhu hodnot na všech důležitých vodičích při úspěšném provedení kompletního zarovnaného zápisu jednoho slova s hodnotou `$5` na adresu `$E0000000` (transakce byla iniciována prováděním instrukce `mov [$E0000000], eax` v CPU, když byla v registru `eax` hodnota `$5`; transakce byla přijmuta uvedeným PCI/ I^2C HBA). Nakreslete průběh pro každý vodič sběrnice PCI zvlášť, přičemž seskupit do jednoho smíte jen vodiče, které mají v celém průběhu transakce shodnou hodnotu. K vašemu obrázku napište krátkou legendu, tj. pro každý použitý vodič vysvětlete jeho význam, a uveďte, které zařízení v uvedené konfiguraci „generuje“ signál na kterém vodiči.

Otázka č. 2 (X)

Předpokládejte, že implementujeme jádro nového OS, který má běžet na uvedeném počítači a bootovat z uvedeného flash disku a má podporovat práci s veškerými instalovanými zařízeními. Navrhněte (nakreslete obrázek) jakým způsobem bychom v takové situaci jádro OS strukturovali. Zaměřte se na podporu aplikačního souborového API, a aplikačního API zjišťování aktuální souřadnice kurzoru myši. Předpokládejte, že dotek na připojeném displeji má být ekvivalentní přesunutí kurzoru myši na souřadnice doteku.

Pokud budete kód jádra OS rozdělovat do více modulů, tak každý takový modul označte a stručně popište jeho funkci, a vyznačte všechny ostatní moduly a části OS, se kterými bude každý takový modul komunikovat. Můžete předpokládat, že náš OS poběží jen na procesorové architektuře x86. Na druhou stranu ale předpokládejte, že bychom chtěli být schopni do OS maximálně jednoduše přidávat podporu pro další zařízení a souborové systémy. Do vašeho návrhu zahrňte i podporu pro mechanizmus potřebný pro vyřešení problému níže uvedené otázky č. 3.

Otázka č. 3 (X)

Žádný z výše uvedených řadičů připojených na uvedené PCI sběrnici nemá v sobě napevno nastavenou žádnou adresu registrů svého HCI v paměťovém ani I/O adresovém prostoru. Přesto bude OS schopen s těmito řadiči komunikovat. Detailně vysvětlete, jak je to možné, resp. co vše, a jakým způsobem, a kdy pro to musí OS udělat. Pokud kvůli tomu musí OS s „někým“ komunikovat, tak v kontextu uvedeného počítače vysvětlete, kdy, proč, a s kým.

Otázka č. 4

Předpokládejte, že v typickém OS máme ovladač zařízení A (např. AHCI řadič disků) přímo připojeného na sběrnici PCI Express, a jiný ovladač zařízení B (např. externí USB disk) přímo připojeného na sběrnici USB. Liší se nějak principiálně způsob komunikace se zařízením A a se zařízením B z pohledu jejich ovladačů, když jsou obě sběrnice PCIe i USB sériové a data se po nich přenášejí ve formě paketů? Vysvětlete proč!

Otázka č. 5

Zapište v šestnáctkové soustavě hodnotu 32-bitové proměnné `i` typu `Longint` po provedení níže uvedeného kódu v Pascalu, když víme, že se kód bude překládat pro little endian platformu:

```
i := -20; i := ($FFFF xor (i shr 2))
                and not($50 shl 16);
```

Otázka č. 6

Předpokládejte, že v typickém OS implementujete obsluhu nějakého maskovatelného přerušení (vznikajícího jako důsledek IRQ požadavku nějakého zařízení). Je potřeba si na začátku obsluhy přerušení zapamatovat stav nějakých (všech) registrů procesoru a obnovit tento jejich původní obsah před provedením instrukce návratu z přerušení? Platí to i pro příznaky v příznakovém registru? Pokud ano, tak jaké? Vysvětlete proč! Důkladně si rozmyslete, v jakých všech situacích může k vyvolání obsluhy přerušení dojít.

Otázka č. 7

V operačním systému s podporou pro vícevláknové zpracování s preemptivním přepínáním vláken a s podporou pouze pro jednoprocesorové systémy chceme naimplementovat proceduru `TerminateSelf`, která má ukončit volající vlákno. Zapište implementaci takové procedury v Pascalu (stručně popište datové struktury a procedury a funkce, které ve své implementaci využijete). Také vysvětlete, co se bude dít, když v čase volání `TerminateSelf` nejsou v systému žádná vlákna ve stavu `Ready-to-Run`. Je to problém? Může taková situace vůbec nastat? Vysvětlete proč!

Otázka č. 8

Předpokládejte, že je naším úkolem v Pascalu naprogramovat tzv. CLR (Common Language Runtime), tedy samotnou VM .NETu. Napište v Pascalu zjednodušený základ takové VM s harvardskou a se zásobníkovou registrovou architekturou (všechny registry obsahují 32-bit signed celá čísla) jako interpret CIL (Common Intermediate Language) kódu pro následující 4 instrukce (vše jednobytový opcode následovaný případnými argumenty):

- `ble` (opcode `0x3E`) – podmíněný skok: instrukce odebere ze zásobníku dvě hodnoty, a pokud je druhá odebraná menší nebo rovna první odebrané, tak se provede skok na adresu, která je argumentem instrukce. Za opcodem vždy následují 4 byty, kde je v little endian pořadí zapsána cílová adresa skoku jako celé beznaménkové číslo.
- `ldc.i4.0` (opcode `0x16`) – load constant `0` (bez arg.)
- `ldc.i4.2` (opcode `0x18`) – load constant 2 (bez arg.)
- `mul` (opcode `0x5A`) – násobení (bez arg.)

K dispozici máte implementaci zásobníku formou jednosměrně vázaného seznamu `Longint` hodnot, viz kód níže. Prázdný zásobník je reprezentován hodnotou `nil`. Procedura `Push` vkládá novou hodnotu na vršek zásobníku daný parametrem `top`, a do `top` nastaví nový vršek zásobníku. Funkce `Pop` vrací hodnotu z vršku zásobníku a vršek v `top` aktualizuje na další položku v zásobníku nebo `nil`. Předpokládejte, že CIL kód k provedení je uložen v globální proměnné `cil`, která reprezentuje kódový adresový prostor VM, a že první má být provedena instrukce na adrese (indexu) `0`.

```
var
   cil : array[0..4294967295] of Byte;
type
   PReg = ^TReg;
   TReg = record
      value : Longint;
      next : PReg;
   end;
procedure Push(var top : PReg;
               value : Longint);
function Pop(var top : PReg) : Longint;
```

Otázka č. 9

Vysvětlete, co je, jak se chová, kdo implementuje, a k čemu se používá tzv. operace *Compare and Swap*.

Otázka č. 10

Předpokládejte, že máme následující typ, který nám umožňuje ve formě jednosměrně vázaného seznamu reprezentovat dlouhá celá beznaménková čísla po jednotlivých 32 bitech (tedy n položkový seznam nám reprezentuje jedno n*32-bitové číslo, kdy první položka seznamu reprezentuje 32 bitů s nejnižší váhou, poslední položka [která má v `next` hodnotu `nil`] reprezentuje 32 bitů s nejvyšší váhou; pokud je n rovno `0` [tedy „číslo“ je jen hodnota `nil`], tak dané číslo reprezentuje hodnotu `0`):

```
type
   PPart = ^TPart;
   TPart = record
      next : PPart;
      val : Longword;
   end;
```

Předpokládejte, že položka `next` leží na offsetu `0` od začátku záznamu `TPart`, položka `val` leží na offsetu 4 od začátku záznamu.

Úloha: Čistě v assembleru níže uvedeného 32-bitového procesoru naimplementujte kompletní níže uvedenou proceduru `Add`, která má k dlouhému číslu `a` přičíst dlouhé číslo `b`; pro jednoduchost předpokládejte, že `a` i `b` mají stejný počet položek, dále předpokládejte variantu Cčkové volací konvence (parametry se předávají na zásobníku zprava doleva, parametry odstraňuje volající):

```
procedure Add(a : PPart; b : PPart);
```

Instrukční sada: Kód implementujte pro počítač s variantou 32-bitového procesoru Intel 80386 běžícího v 32-bitovém režimu s vypnutou podporou pro stránkování (virtuální i fyzický adresový prostor procesoru má šířku 32-bitů, virtuální adresa se přímo rovná adrese fyzické). Procesor má obecnou registrovou architekturu, a mimo jiné 7 obecných 32-bitových registrů `EAX`, `EBX`, `ECX`, `EDX`, `EBP`, `EDI`, `ESI`, dále má 32-bitový registr `ESP` (stack pointer), a 32-bitový registr `EIP` (program counter), a příznakový registr se všemi běžnými příznaky. Můžete využít instrukce `MOV`, `ADD`, `ADC`, `CLC` (clear carry), `STC` (set carry), `CMP`, `JZ`, `JNZ`, `JMP`, `PUSH`, `POP`, `PUSHF`, `POPF`, `RET`, které mají všechny standardní chování. Instrukce ovlivňují příznaky standardním způsobem, aritmetické instrukce jsou dvouadresové. Pro zápis vašeho kódu využijte běžné Intel syntaxe, tedy: cílový operand instrukce je vždy nejvíce vlevo; immediate hodnota je libovolné číslo bez prefixu; hodnota v hranatých závorkách znamená variantu instrukce s operandem typu adresa, tj. např. `[x]` znamená hodnotu operandu na adrese `x`; běžné instrukce procesoru 80386 mají i variantu, kde se adresa operandu spočítá jednoduchým výpočtem, a tedy např. zápis `[x+y]` znamená operand ležící na adrese, která vznikne součtem hodnot `x` a `y`, kde `x` i `y` mohou být libovolný registr, pouze `y` může být i hodnota immediate, u instrukcí s více operandy smí být pouze jeden adresa v `[]`. Cíl skoku si můžete v assembleru označit jako label podobně jako v Pascalu, tedy libovolné jméno s dvojtečkou. ***Př.:*** pokud je v registru `EDI` hodnota `$00AA0050`:

```
něco:                  { následující instrukce mov je cílem skoku }
   mov eax, [edi+4]    { načtení 4B hodnoty z adresy $00AA0054
                         a její uložení do registru EAX }
   jmp něco            { proveď skok na adresu instrukce za
                         lablem něco, tj. zde na instrukci mov }
```