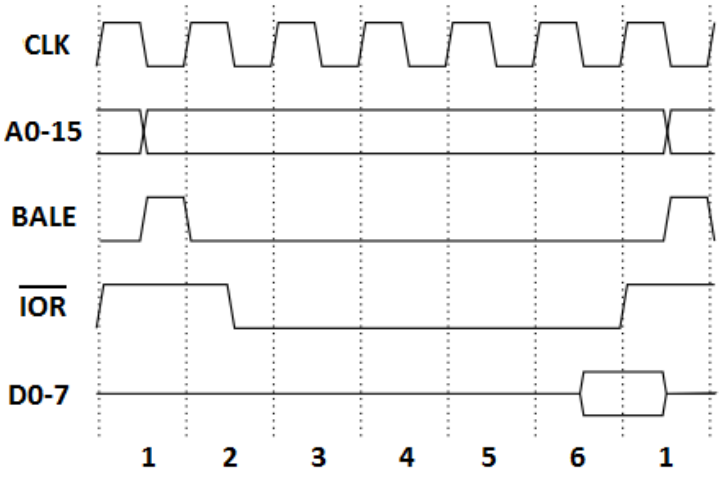
Odpovědi pište na zvláštní odpovědní list s vaším jménem a fotografií. Pokud budete odevzdávat více než jeden list s řešením, tak se na 2. a další listy nezapomeňte podepsat. Do zápatí všech listů vždy napište i/N (kde i je číslo listu, N je celkový počet odevzdaných listů).

**Společná část pro otázky označené X**

Předpokládejte, že máme počítač s 16-bitovým procesorem Intel 8088, kde je systémovou sběrnicí 8-bitová varianta sběrnice ISA (8-bitová datová cesta, podpora pro 20-bitový paměťový a 16-bitový I/O adresový prostor). Tato sběrnice má následující podobu čtecího cyklu z I/O adresového prostoru (můžete předpokládat pevnou délku transakce, informace na sběrnici musí být platné při úrovni 0 hodinového signálu, a přenášená data se mohou na sběrnici objevit již od 2. taktu, nicméně vždy musí být přítomna minimálně v 6. taktu [jak ukazuje obrázek]):




---

**Otázka č. 1 za 0,5 bodu (X)**

Popište a vysvětlete význam jednotlivých signálů takovéto sběrnice.

---

**Otázka č. 2 za 1,5 bodu (X)**

Předpokládejte, že pro uvedený počítač chceme navrhnout řadič klávesnice:

***Řadič klávesnice*** má mít jeden 8-bitový port-mapped registr na adrese 60h. Čtením z toho registru získáme kód naposledy stisknuté klávesy, po přečtení se kód klávesy nastaví na 0 (hodnota 0 v registru 60h značí, že od posledního čtení nebyla žádná klávesa stisknuta).

Řadič klávesnice chceme vyrobit formou naprogramování 8-bitového MCU s harvardskou architekturou:

***MCU*** má zabudovaný řadič synchronní sériové sběrnice (piny procesoru č. 48 a 49), a řadič 5 krát 8 (tj. celkem 40) nezávislých GPIO linek (programově ovladatelných digitálních vstupně/výstupních pinů procesoru číslovaných 0 až 39). Pro každou z těchto pěti osmic GPIO pinů má řadič vyhrazenu jednu 8-bitovou výstupní paměť SRAM. HCI tohoto řadiče GPIO tvoří šest 8-bitových registrů:

- R/W registr MODE (na adrese 100h) slouží k přepínání mezi režimy vstup nebo výstup pro celé osmice GPIO linek (bit 0 určuje režim 1. osmice, bit 1 určuje režim 2. osmice, atd. pro 3. až 5. osmici: hodnota 0 určitého bitu [počáteční hodnota po restartu MCU] = režim vstup pro danou osmici, hodnota 1 = režim výstup). Pokud je daná osmice v režimu vstup, tak je všech 8 pinů dané osmice odpojených (floating) ze strany MCU. Pokud je daná osmice v režimu výstup, tak je všech 8 pinů osmice připojených na kladné napěti = 1 nebo zem = 0 (dle obsahu registru IO*n* pro danou osmici *n*, viz dále).
- R/W registr IO1 (na adrese 101h) slouží k ovládání 1. osmice pinů, tj. ke čtení/zápisu 8 jednobitových hodnot z/do pinů 0 (bit 0) až 7 (bit 7).
  Čtením z tohoto registru (jen pokud je osmice v režimu vstupu) získáme aktuální stav na všech pinech 0 až 7 (pokud je pin vně MCU zapojen na kladné napětí je v daném bitu hodnota 1, pro zem je v bitu hodnota 0, pro nepřipojený pin je v bitu náhodná hodnota).
  Zápisem do tohoto registru (jak v režimu výstupu tak i v režimu vstupu této 1. osmice) se mění obsah 1 bytové výstupní paměti řadiče pro tuto osmici. V režimu výstupu je obsah výstupní paměti této osmice „vysílan" MCU na pinech 0 až 7.
- Podobně fungují i registry IO2 (na adrese 102h) pro piny 8 až 15 (opět bit 0 = pin 8, až bit 7 = pin 15), IO3 (na 103h) pro piny 16 až 23, IO4 (na 104h) pro piny 24 až 31, a IO5 (na 105h) pro piny 32 až 39.

***Úloha:***

Předpokládejte, že uvedený MCU připojíme následujícím způsobem ke sběrnici ISA: piny 0-7 na vodiče D0-D7, piny 8-23 na vodiče A0-A15, pin 24 na vodič BALE, pin 25 na vodič IOR, pin 26 na vodič CLK. Dále piny 48 a 49 (sériové sběrnice) budou připojeny na CLK a DATA signály sériové sběrnice PS/2 konektoru pro připojení klávesnice.

Pro toto zapojení napište v Pascalu program firmware pro takový MCU tak, aby se celou dobu svého běhu choval jako výše popsaný řadič klávesnice. Předpokládejte, že v programu bude globální proměnná `stav : Byte`, do které nám obsluha přerušení sériové sběrnice (zajišťující pro nás komunikaci s vlastní klávesnicí) vždy uloží kód naposledy stisknuté klávesy *(takovou obsluhu přerušení ale do programu doplníme až později, nyní ji **NE**programujte)*.

Dále předpokládejte, že taktovací frekvence MCU je pro vaše potřeby dostatečně vysoká (řádově větší než frekvence sběrnice ISA). Pro čekání na změnu stavu nějakého signálního vodiče sběrnice ISA používejte aktivní čekání (pollování).

***Doporučení:*** Pro větší přehlednost si v programu zaveďte pojmenované konstanty pro bity reprezentující hlavní kontrolní signály.

---

**Otázka č. 3**

Popište a vysvětlete, co to je *interpret* programovacího jazyka. Zvláště vysvětlete jaké výhody a nevýhody má použití interpretu místo použití překladače.

---

**Otázka č. 4**

Předpokládejte operační systém s podporou pro vícevláknové zpracování a s preemptivním přepínám vláken. Jaké další (pokud nějaké) vlastnosti musí takový OS splňovat, aby v něm mohlo dojít k problému zvanému *priority inversion*? Na příkladu pak uveďte, kdy k takovému problému dochází, a navrhněte jeho řešení.

---

**Otázka č. 5**

Mějme dva různé typy moderních procesorů P1 a P2, které mají zcela shodnou architekturu i rychlost, a liší se pouze velikostí cache (P2 má 2-krát větší cache než P1). Napište v Pascalu příklad programu (algoritmu), který by mohl na procesoru P1 běžet zásadně pomaleji než na P2 (při dnes běžných velikostech a funkci procesorových pamětí cache).

---

**Otázka č. 6**

Předpokládejte, že máme počítač s 64-bitovým procesorem Intel i5-2300 taktovaným na frekvenci 3,1 GHz. Procesor má v sobě zabudovaný řadič pamětí typu DDR3-1066 – tato paměťová 64-bitová paralelní sběrnice je taktována na 533 MHz, podporuje burst přenosy, a v jednom taktu na ní proběhnou 2 kompletní transakce (1. přenos při náběžné hraně hodinového signálu [z `0` na `1`], a 2. přenos při sestupné hraně [z `1` na `0`]). Dále má v sobě procesor zabudován PCI Express 2.0 root complex s 16 lanes vedoucími z procesoru. Počítejte s tím, že typicky uváděná rychlost PCIe sběrnice je čistou rychlostí na její nejnižší fyzické úrovni, a že `Memory Write` PCIe paket s 32 b cílovou adresou má celkem 16 b preambuli, 18 B hlavičky, a maximálně 4096 B dlouhý payload. Interní procesorová sběrnice propojující tyto řadiče a samotné procesorové jádro je rychlejší než všechny uvedené sběrnice.
Na základní desce je na jedné PCIe lane připojen PCIe 1.0/PCI bridge, ze kterého je do 4 slotů vyvedena běžná varianta PCI sběrnice. Ve 2. PCI slotu je zapojen AHCI SATA 1.0 HBA s podporou DMA bus masteringu, a k němu je připojen pevný disk Seagate 7200.14 1000GB s adv. format. V tomto systému budeme z pevného disku do operační paměti (od adresy `$02AFC000` dál) načítat obsah 1 GB souboru, jehož data jsou souvisle uložena od sektoru číslo `1899`. Víme, že se přenos uskuteční formou DMA. Pokud by v systému neprobíhaly žádné další přenosy dat, a využívaly by se maximální délky burst přenosů, jak dlouho bude přibližně trvat načtení uvedeného souboru do paměti? Zapište a vysvětlete i postup výpočtu!

---

**Otázka č. 7**

Předpokládejte, že vytváříme embedded systém postavený na procesoru Intel 80486 kompatibilním (32-bitové CPU) a na operačním systému MS-DOS, jehož kód běží v 16-bitovém režimu daného procesoru (16-bitové registry, 20-bitový paměťový adresový prostor přímo mapovaný na spodní 1 MB fyzického 32-bitového adresového prostoru CPU). OS tedy o paměti nad 1 MB neví a ani se o ni nestará. Veškeré API funkce OS, které pracují s nějakými daty aplikací (čtení/zápis do souboru, apod.), mají jako argument určující adresu takového bufferu 20-bitový pointer. OS nemá podporu pro vícevláknové zpracování.
Pro tento systém ale plánujeme programovat 32-bitové aplikace, které vždy po spuštění přepnou procesor do 32-bitového režimu (32-bitové registry, 32-bitový paměťový adresový prostor). Každá taková aplikace bude ale využívat API uvedeného OS MS-DOS (mimo jiné pro veškeré čtení a zápisy z/do souborů na disku), a bude tedy muset vždy CPU přepnout zpět do 16-bitového režimu před voláním nějaké API funkce OS. Očekáváme, že aplikace budou běžně číst data ze souborů do datových struktur dynamicky alokovaných na haldě, a očekáváme, že při typickém běhu dosáhne velikost haldy aplikace až 500 MB – tedy bude třeba vyřešit, jak se budou přenášet data mezi OS a libovolnou částí haldy.
Bylo by možné nějak zařídit, abychom tato specifika komunikace mezi naším 32-bitovým kódem a 16-bitovým kódem OS nemuseli opakovaně programovat v každé takové aplikaci znovu? Pokud ne, tak vysvětlete proč. Pokud ano, tak vysvětlete, jak toho docílíte. V obou případech i navrhněte řešení problému s velkou haldou a voláním OS.

---

**Otázka č. 8**

Naprogramujte v Pascalu proceduru `Zkratka` s následující deklarací:

```
type
    PUtf32 = ^Longword;
procedure Zkratka(
    text : PUtf32; zkr : PUtf32);
```

Parametr `text` ukazuje na null-terminated UTF-32 textový řetězec. Procedura má z tohoto řetězce vzít kompletní první písmeno přidat k němu znak tečka, a překopírovat je na místo určené parametrem `zkr`. Parametr `zkr` ukazuje na předalokované dostatečně velké místo v paměti, kam se má uložit výsledek procedury `Zkratka` jako null-terminated UTF-32 řetězec. Tedy pokud bude proměnná `text` ukazovat např. na text „`Čeněk`“ v libovolné validní UTF-32 reprezentaci, tak po zavolání procedury `Zkratka` má být na adrese dané parametrem `zkr` uložen text „`Č.`“. Pokud `text` ukazuje na prázdný řetězec, tak ve `zkr` má být uložen také prázdný řetězec. Předpokládejte, že typ `Longword` je 32-bitové celé číslo bez znaménka, typ `Char` má velikost 8 bitů, znakové konstanty ve zdrojovém kódu (v apostrofech) překladač chápe jako znak v kódování ISO 8859-2, a znak tečka je podporován již základním kódováním ASCII. Pokud byste od RTL potřebovali nějaké funkce pro získání informací o jednotlivých UTF-32 nebo Unicode znacích, tak si takové funkce zadeklarujte (ale neimplementujte), a popište jejich očekávané chování.

---

**Otázka č. 9**

Předpokládejte, že implementujete jádro operačního systému s kooperativním přepínáním vláken s jednoduchým round robin plánovačem. Naimplementujte v Pascalu API funkci `Join` plánovače takového OS. Stručně popište a vysvětlete všechny datové struktury, globální proměnné, a procedury a funkce, které budete ve vaší implementaci používat.

---

**Otázka č. 10**

Předpokládejte, že máme na USB flash disku nainstalovaný nějaký běžný OS. Tuto „flešku“ zapojíme do USB konektoru vypnutého notebooku IBM PC kompatibilního s 64-bit CPU AMD Phenom II. Po zapnutí počítače dojde k nabootování do našeho OS. Popište a vysvětlete, co se děje, a jaký kód CPU zpracovává, v průběhu celého procesu bootování, tj. od zapnutí napájení až do začátku běhu kódu shellu OS.