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).

Otázka č. 1

Co nejefektivněji rozhodněte, pro která z následujících 128-bitových čísel platí, že hodnota 117. bitu (bráno běžným počítáním bitů, tedy bit se 117. váhou, počítáno od 0) se rovná hodnotě 1. bitu, **a současně** hodnota 116. bitu se rovná hodnotě 0. bitu. Stručně napište, jak jste postupovali.

(a) `$eb9183ca4b06353a7148a3491aef1fb5`
(b) `$c859f9a1890767cb27f2308213986241`
(c) `$bb148b6f9187a7e55d0045ceb9daf116`
(d) `$21bda43a691aa8f268c543bbd52d8c65`
(e) `$02eaffe5b374818201094f28142bbbae`
(f) `$4bfd8aea116323c731f7966fbdafbd2d`
(g) `$1465a1dcf6a12a98aa371cfd526fdc96`
(h) `$a9791e19a085ae3d12a1d8530db80ef3`
(j) `$429c4aa94f76091541ccd02406e8bab9`
(k) `$fda1fcc1e53322d1d8a9ef8696cc8969`

Společná část pro otázky označené X

Předpokládejte následující program zapsaný v jazyce Pascal (typ `longint` je 32-bitový celočíselný se znaménkem):

```
1   program MyPrg;
2   var
3     a : longint;
4     b : longint;
5     c : longint;
6
7   procedure Compute;
8   begin
9     c := a + b;
10    c := c div (a – b);
11  end;
12
13  begin
14    ReadLn(a);
15    ReadLn(b);
16    Compute;
17    WriteLn(c);
18  end.
```

Dále předpokládejte nějaký 32-bitový procesor **s obecnou registrovou architekturou** (tedy **není** to CPU ani s akumulátorovou, ani s registrovou zásobníkovou [stack machine] architekturou) a s podporou pro hardwarové celočíselné dělení – dostačuje nějaká obecná představa procesoru, jehož instrukční sada by rámcově odpovídala typickému návrhu takových procesorů. Tedy můžete využít instrukce nějaké reálné procesorové architektury (např. x86/IA-32 nebo MIPS), nicméně stejně tak je plně dostačující, pokud si nějakou přiměřeně realistickou instrukční sadu vymyslíte sami. Předpokládejte, že uvedený procesor má podporu pro běh moderních desktopových operačních systémů jako Linux nebo Windows.

Otázka č. 2 (X)

Předpokládejte, že takový procesor používá běžný systém obsluhy přerušení a faultů. Vysvětlete v tomto kontextu, co je tabulka vektorů přerušení. Popište, jak by na uvedeném procesoru mohla vypadat (vhodné konstanty si vymyslete), tj. jaká by byla její struktura a obsah. Kdo přesně bude vyplňovat a kdo číst její obsah?

Otázka č. 3 (X)

Zapište v assembleru takového procesoru posloupnost instrukcí, do které by se nejspíše přeložila celá procedura `Compute` nějakým typickým překladačem Pascalu (kód zapište včetně prologu a epilogu procedury).
Předpokládejte, že překladač Pascalu nedělá žádné optimalizace kódu.
Adresy proměnných si libovolně **vhodně zvolte** – do svého řešení **napište** nějaká vhodná **konkrétní čísla** (a jejich vhodnost zdůvodněte).
Pro každou použitou instrukci napište **stručný** popis jejího chování.

Otázka č. 4 (X)

Pokud výše uvedený přeložený Pascalový program spustíme a na standardní vstup zadáme:

```
42
42
```

tak vypíše na standardní výstup:

```
Runtime error 200 at $0040143C
  $0040143C  COMPUTE,  line 10 of MyPrg.pas
  $004014A6  main,  line 16 of MyPrg.pas
  $00408271
```

Vysvětlete kdo (jaký kód) nejspíše vypisuje text uvedené chybové hlášky "`Runtime error ...`".
Dále se vraťte ke svému řešení **otázky 3**, a na něm vysvětlete, jak je přesně zařízeno, že procesor bude vykonávat vámi uvedený strojový kód, a přesto při detekci dělení nulou dojde k výpisu chybové hlášky "`Runtime error ...`". Jaký všechen kód bude procesor při dělení nulou vykonávat? Stačí vyjmenovat a vysvětlit celou posloupnost jednotlivých částí operačního systému, uvedeného programu, atd., které se na ošetření dělení nulou budou v typické situaci podílet. **Odpovídejte v kontextu nějakého moderního desktopového OS (s podobnými vlastnostmi jako Linux nebo Windows).**

Otázka č. 5 (X)

Detailně vysvětlete, odkud se za běhu programu asi vezme informace o jménu procedury, o čísle řádku programu, na kterém došlo k chybě, a o jménu zdrojového souboru – viz chybový výpis v **otázce 4** výše. Kde jsou tyto informace asi uložené? Můžeme předpokládat, že tyto informace budou za běhu k dispozici vždy, nezávisle na způsobu překladu původního programu, nebo se může stát, že bychom měli výsledný spustitelný soubor, při jehož spuštění a při vzniku stejné chyby jako v **otázce 4** tak by se vypsala pouze adresa instrukce, která chybu způsobila?

Otázka č. 6

Předpokládejte, že implementujeme RTL jazyka Pascal a naším úkolem je v Pascalu naprogramovat standardní funkci `IntToHex`, která má předané číslo převést do šestnáctkové soustavy (stejně jako fce ve Free Pascalu, jak ji známě z přednášek). Nicméně pro jednoduchost předpokládejte, že má funkce pouze jeden argument, a že výsledek má vždy tolik platných číslic kolik odpovídá maximálního rozsahu zvoleného vstupního typu (tedy číslo je zleva dorovnané nulami). Napište implementaci takové funkce včetně deklarace. Očekávejte, že typ `char` je 1 bytový, a znak v něm uložený je v 8-bitovém rozšíření kódování ASCII.

Otázka č. 7

Předpokládejte nějakou běžnou implementaci jednosměrně vázaného seznamu. Pokud je jeden seznam sdílený 2 vlákny v jednom programu běžícím na nějakém typickém operačním systému s preemptivním přepínám vláken (např. OS Windows 10), a pokud vlákna nepoužívají žádná synchronizační primitiva pro ochranu seznamu, může dojít k nějakému naplánování vláken, že při vládání nebo odebírání položek ze seznamu dojde k race condition, která způsobí, že bude seznam v nekonzistentním stavu vzhledem k pohledu alespoň jednoho z vláken?
Pokud ano, tak napište konkrétní proložení příkazů vláken, kdy by k takové situaci mohlo dojít a vznikající problém vysvětlete. Pokud ne, tak detailně vysvětlete, proč k race condition nemůže dojít.

Otázka č. 8

Předpokládejte, že chceme reálné číslo `1021,75` uložit do Pascalové proměnné ve standardním formátu single. Typ `single` je 32-bitové floating-point číslo dle standardu IEEE 754, tj. mantisa je normalizována se skrytou 1 a zabírá spodních 23 bitů, pak následuje 8-bitový exponent uložený ve formátu s posunem [bias] +127, a poslední bit, tedy `MSb`, je znaménkový bit.
Nyní program obsahující takovou proměnnou spustíme na počítači s 32-bitovým little-endian CPU, do proměnné uložíme uvedenou hodnotu `1021,75`, a zjistili jsme, že je proměnná uložená na adrese `0x0E07F410`. Napište v šestnáctkové soustavě hodnotu každého bytu paměti, ve kterém bude uložená nějaká část proměnné.

Otázka č. 9

Naprogramujte v Pascalu program, který jako 1. argument na příkazové řádce dostane jméno souboru s binárním obrazem nějakého disku a zpracuje ho, viz informace dále.
Informaci o argumentech předaných aplikaci na příkazové řádce lze zjistit pomocí standardní Pascalové funkce `ParamStr`, která vrací hodnotu konkrétního argumentu na příkazové řádce jako textový řetězec:

```
function ParamStr(argNum : longint) : string;
```

Tedy např. volání `ParamStr(1)` vrací hodnotu 1. argumentu předaného na příkazové řádce.
Dále předpokládejte, že předaný obraz disku obsahuje kopii obsahu disku s 512 KiB sektory, a že původní disk je naformátován způsobem běžným na desktopových PC, tedy že 0. sektor celého disku slouží jako tzv. *master boot record (MBR)*, a kromě jiného obsahuje též informace o jednotlivých oddílech (partitions), na které se je disk dále rozdělen – specifikaci struktury MBR a jeho jednotlivých položek převzatou z Wikipedie najdete ***v 1. příloze*** tohoto zadání (víme, že v předaném obrazu disku je použitá varianta „*classical generic MBR*“).
Úkolem vašeho programu je pro každý oddíl, který je v MBR označený jako aktivní/bootovatelný (tj. položka konkrétního oddílu v MBR má nastavený bit „*active or bootable*“) vypsat v desítkové soustavě C/H/S adresu jeho prvního a posledního sektoru.

1. příklad výstupu programu:
   Pokud by např. např. byl bootovatelný pouze 2. oddíl (mezi adresami [sektory] 1/2/3 a 10/4/5) a 4. oddíl (mezi adresami [sektory] 20/6/7 a 40/8/9), tak má váš program vypsat následující text:

   ```
   2: 1/2/3 - 10/4/5
   4: 20/6/7 - 40/8/9
   ```

2. příklad výstupu programu:
   Pokud bychom vašemu programu předali obraz disku, který by obsahoval MBR ***z 2. přílohy*** tohoto zadání, tak by měl váš program vypsat:

   ```
   1: 0/1/1 - 1/254/63
   ```

Otázka č. 10

Předpokládejte následující deklarace:

```
type
     PByte = ^byte;
     PInfo = ^TInfo;
     TInfo = record
       Used : byte;
       Buffer : PByte;
       Next : PInfo;
     end;

var x : array[0..2] of TInfo
```

Nakreslete, jak bude asi celá proměnná `x` vypadat v paměti po přeložení programu nějakým běžným překladačem Pascalu, a jeho spuštění. Pro každou složku každého záznamu v proměnné `x` napište její offset od bázové adresy proměnné `x`. Napište stručné zdůvodnění vaší odpovědi.