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

Předpokládejte nějaký běžný moderní operační systém jako Linux nebo Windows. Napište, jaké hlavní části budou součástí stavu (kontextu) běžného procesu, a pro každou stručně vysvětlete, co znamená.

Otázka č. 2

Předpokládejte, že implementujeme I²C ambient light sensor, který komunikuje stejným protokolem, jaký používá sensor ALS-PDIC17-57B/TR8 společnosti Everlight (datasheet je přílohou tohoto zadání – pro tuto otázku je důležitá hlavně strana 11). Při návrhu našeho sensoru jsme ale narazili na problém, který vzniká, když master s naším zařízením komunikuje s využitím taktovací frekvence `100 kHz`. Pokud totiž přijímáme transakci čtení 16-bitového slova z našeho sensoru, tak po úspěšném příjmu adresového bytu a jeho potvrzení pozitivním acknowledgementem nejsme nikdy schopni začít posílat první byte z ADC registru dříve než za `0,1` milisekundy. Detailně vysvětlete, zda (a případně jak) nám I²C sběrnice dává nějakou možnost řešení daného problému.

Otázka č. 3

Předpokládejte následující deklarace (kde typ `longword` je 32-bitový celočíselný bezznaménkový, typ `word` je 16-bitový celočíselný bezznaménkový):

```
type
  PLongword = ^longword;
  PWord = ^word;
procedure Conv(src : PWord; dst : PLongword);
```

Napište ve Free Pascalu implementaci procedury `Conv` tak, aby převedla vstupní textový null-terminated řetězec `src` z kódování UTF-16 LE do kódování UTF-32 LE a výsledné znaky UTF-32 LE null-terminated řetězce uložila do paměti na místo, kam ukazuje argument `dst` (předpokládejte, že váš kód poběží pouze na little-endian platformách, a že na místě, kam ukazuje proměnná `dst`, je dostatek nevyužité paměti). Počítejte s tím, že znaky s Unicode kódem z rozsahu `$00D800` až `$00DFFF` se nemohou nikdy vyskytnout v platném textovém řetězci, a že mohou být konkrétním Unicode kódováním použity pro zakódování jiných znaků. Dále víme, že kódům větším než `$10FFFF` není přidělen žádný platný znak, a že Unicode znaky z rozsahu `$010000` až `$10FFFF` se v UTF-16 kódují následujícím způsobem:

(1) Od kódu znaku se odečte hodnota `$010000`, a výsledné 20-bitové číslo se rozdělí na dvě 10-bitové části, které se zakódují dle následujících pravidel.

(2) Nejvyšších 10-bitů 20-bitové hodnoty se uloží do nejnižších 10 bitů prvního 16-bitového surrogate znaku (leží na nižší adrese). Horních 6 bitů první surrogate je nastaveno na (vlevo je hodnota bitu 15, vpravo bitu 10):

`1101 10`

(3) Nejnižších 10-bitů 20-bitové hodnoty se uloží do nejnižších 10 bitů druhého 16-bitového surrogate znaku (leží na vyšší adrese). Horních 6 bitů druhé surrogate je nastaveno na (od první surrogate se liší pouze 10. bitem):

`1101 11`

Otázka č. 4

Zakládáme 1. matfyzácký řetězec s rychlým občerstvením *MFC* (*Malostranský Fried Cheese*), kde kromě smažáku bude důležitým prodejním artiklem i bohatá nabídka nápojů od společnosti *Pepsi Co*. Pro prodej nápojů jsme se rozhodli převzít model našeho nejbližšího konkurenta (*KFC*), kde si zákazníci mohou do kelímku natočit libovolné množství vybraného nápoje. Pro naše restaurace tedy vyrábíme podobný postmix s dotekovou obrazovkou (viz obrázek níže) s rozlišením `576x1024` pixelů, kde bude po zapnutí

neustále zobrazena nabídka nápojů (každému odpovídá jedna čtvercová výběrní plocha `180x180` pixelů) – tekutina má z postmixu téct právě tehdy, když se uživatel dotýká prstem obrazovky v místě obrázku nápoje. Do výstupní trysky postmixu je zaveden vstup kohoutkové vody (ovládaný elektromagnetickým ventilem `Vvoda`), vstup vody obohacené o oxid uhličitý tedy CO_2 (ovládaný ventilem `Vsoda`), a 5 trysek ze zásobníků se sirupy jednotlivých příchutí (ventily `V1` až `V5`). Nabídka nápojů bude následující (v závorce uvedena souřadnice levého horního rohu ovládacího čtverce, a číslo ovládacího ventilu): *Pepsi* (`198,60→1`), *Pepsi Light* (`85,280→2`), *7UP* (`310,280→3`), *Mirinda* (`85,490→4`), *Lipton Ice Tea* (`310,490→5`) – pouze ledový čaj se mixuje s čistou vodou, vše ostatní se mixuje se sodou. Navíc je možnost natočit si čistou vodu (`85,700`) a čistou sodu (`310,700`). Pro řízení postmixu používáme 32-bitový µC se zabudovaným řadičem I²C sběrnice, a s řadičem 32 digitálních vstupně výstupních GPIO pinů procesoru (`IO0` až `IO31`). Na pinu `IO0` je připojen ventil `Vvoda`, piny `IO1` až `IO5` → ventily `V1` až `V5`, pin `IO6` → ventil `Vsoda`. Všechny GPIO linky jsou v řadiči implicitně nakonfigurovány jako výstupní a pro změnu jejich stavu již máme připravenu proceduru (LSbit reprezentuje stav `IO0`, MSbit reprezentuje stav `IO31`):

```
procedure SetGpioOutputs(pins : longword);
```

Na I²C sběrnici je připojen řadič dotekové vrstvy obrazovky. V Pascalu již máme naprogramovánu funkci:

```
function GetTouchInfo(
  var x : longword; var y : longword) : boolean;
```

která pomocí I²C řadiče přečte poslední zaznamenanou událost na dotekové vrstvě (v parametrech `x` a `y` vrací souřadnice události; návratová hodnota: `true` = zaznamenán dotek, `false` = zaznamenána ztráta doteku). Naprogramujte v Pascalu firmware pro uvedený počítač tak, aby se choval přesně podle specifikace postmixu uvedené výše – program napište co nejjednodušeji pollováním GPIO řadiče i dotekové vrstvy s využitím výše uvedených funkcí.

Společná část pro otázky označené X

Předpokládejte, že známe specifikaci varianty virtual machine vycházející z CLR (Common Language Runtime) = standardní VM .NETu. Naše VM je stroj s **harvardskou** a se **zásobníkovou registrovou** architekturou (všechny registry obsahují **32-bit signed** celá čísla, velikost registrového zásobníku je neomezená). Strojový kód pro tuto VM budeme nazývat tzv. CIL kód (Common Intermediate Language). Víme, že v CIL kódu i v samotné virtual machine jsou všechna data uložena jako **little-endian**, a že instrukční sada VM obsahuje minimálně tyto instrukce (všechny mají jednobytový opcode následovaný případnými argumenty):

- `ldsfld` (opcode `0x7E`) – load static field = load z globální proměnné, jejíž adresa je argumentem instrukce
- `stsfld` (opcode `0x80`) – store static field = store do globální proměnné, jejíž adresa je argumentem instrukce
- `call` (opcode `0x28`) – volání procedury nebo funkce (argumenty se předávají zleva doprava na registrovém zásobníku a **odstraňuje je volaný**, návratová hodnota se předává na vrcholu registrového zásobníku)
- `ret` (opcode `0x2A`) – návrat z podprogramu (bez ex. arg.)
- `add` (opcode `0x58`) – sčítání (bez explicitních argumentů)
- `mul` (opcode `0x5A`) – násobení (bez explicitních argumentů)
- `ble` (opcode `0x31`) – relativní podmíněný skok branch if less than or equal to: 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 následuje 1 byte reprezentují offset skoku vzhledem k prvnímu bytu instrukce následující za `ble`.

Otázka č. 5 (X)

Napište v Pascalu bez použití inline assembleru kód procedury (i s deklarací), která by mohla být běžným překladačem přeložena do níže uvedeného kódu, který jsme ze spustitelného souboru obsahujícího CIL kód disassemblovali do assembleru CILu:

```
0x20580000 7E 00 00 60 20  ldsfld [0x20600000]
0x20580005 7E 04 00 60 20  ldsfld [0x20600004]
0x2058000a 58              add
0x2058000b 7E 08 00 60 20  ldsfld [0x20600008]
0x20580010 7E 0C 00 60 20  ldsfld [0x2060000C]
0x20580015 7E 10 00 60 20  ldsfld [0x20600010]
0x2058001a 5A              mul
0x2058001b 58              add
0x2058001c 31 26           ble 0x20580044
0x2058001e 7E 08 00 60 20  ldsfld [0x20600008]
0x20580023 7E 00 00 60 20  ldsfld [0x20600000]
0x20580028 7E 04 00 60 20  ldsfld [0x20600004]
0x2058002d 58              add
0x2058002e 7E 00 00 60 20  ldsfld [0x20600000]
0x20580033 7E 04 00 60 20  ldsfld [0x20600004]
0x20580038 5A              mul
0x20580039 28 00 10 58 20  call 0x20581000
0x2058003e 58              add
0x2058003f 80 00 00 60 20  stsfld [0x20600000]
0x20580044 2A              ret
```

Otázka č. 6 (X)

Napište v Pascalu zjednodušený základ výše popsané VM jako interpret CIL kódu pro následující 3 instrukce (ostatní budeme implementovat v budoucnu): `ldsfld`, `stsfld`, `add`.

K dispozici máte implementaci zásobníku formou jednosměrně vázaného seznamu `longint` hodnot, viz kód níže. Prázdný zásobník je reprezentován hodnotou vrcholu rovnou `nil`. Procedura `Push` vkládá novou hodnotu na vrchol zásobníku daný parametrem `top`, a do `top` nastaví nový vrchol zásobníku. Funkce `Pop` vrací hodnotu na vrcholu zásobníku a ukazatel na vrchol v `top` aktualizuje na další položku v zásobníku nebo `nil`.

```
type
  PReg = ^TReg;
  TReg = record
    value : longint;
    next : PReg;
  end;
procedure Push(
  var top : PReg; value : longint); forward;
function Pop(var top : PReg) : longint; forward;
```

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`. Dále máme připravenou globální proměnnou `data` reprezentující datový adresový prostor.

```
var cil : array[0..$40000000] of byte;
    data : array[0..$40000000] of byte;
```

Otázka č. 7

Předpokládejte, že chceme reálné číslo `12,6875` uložit ve *fixed-point* reprezentaci 8.24 do jedné proměnné v našem Pascalovém programu. Jak takovou proměnou v běžném Pascalu deklarujeme? Nyní program obsahující takovou proměnnou spustíme na počítači s 32-bitovým little-endian CPU, do proměnné uložíme uvedenou hodnotu `12,6875`, a zjistili jsme, že je proměnná uložená na adrese `0x02BC4300`. Napište v šestnáctkové soustavě hodnotu každého bytu paměti, ve kterém bude uložená nějaká část proměnné.

Otázka č. 8

Předpokládejte, že v Pascalu programujeme aplikaci, jejíž zdrojový kód je rozdělen do několika samostatných souborů (`a1.pas` až `a3.pas`, kde `a2.pas` obsahuje hlavní tělo programu). Detailně vysvětlete, jak a v jakých krocích vznikne výsledný spustitelný soubor aplikace při použití typického překladače Pascalu. Na jakou instrukci bude ukazovat *entrypoint* takového spustitelného souboru?

Otázka č. 9

Předpokládejte nějaký běžný operační systém jako Linux nebo Windows běžící na procesorech s architekturou x86 (IA-32, tj. 32-bitová varianta Intelovské architektury z běžných počítačů PC). Vyjmenujte všechny důležité stavy, ve kterých se může v takovém OS nacházet nějaké aplikační vlákno. Pro každý uvedený stav vysvětlete, co znamená, a dále vysvětlete, jak se vlákno do takového stavu dostane (např. voláním jakého syscallu jádra).

Otázka č. 10

Napište seznam všech signálních vodičů, které musí mít typická paralelní paměťová sběrnice sloužící k připojení jednoho `2048x16` paměťového modulu typu SRAM. U každého uvedeného vodiče vysvětlete jeho význam.