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

Mikroprocesor MOC 6502 je 8-bitový procesor se 16-bitovou adresovou sběrnicí (paměťovým adresovým prostorem) a s akumulátorovou architekturou. Registr akumulátoru má v assembleru jméno A. V příznakovém registru je příznak Carry (přenos) se standardním chováním označený písmenem C.

Procesor 6502 má následující instrukce:

- LDA `$`*`adresa`*`/#`*`konstanta`* (Load Accumulator) – načtení 8-bitové hodnoty ze zadané 16-bitové adresy (argument instrukce uvozený `$`) *nebo* přímo zadané 8-bitové konstanty (argument instrukce uvozený #) do akumulátoru.
- STA `$`*`adresa`* (Store Accumulator) – uložení hodnoty akumulátoru na zadanou 16-bitovou adresu (argument instrukce)
- ADC `$`*`adresa`*`/#`*`konstanta`* (Add with Carry) – sečtení dvou 8-bitových čísel a příznaku C (druhý operand operace sčítání je argumentem instrukce)
- SBC `$`*`adresa`*`/#`*`konstanta`* (Subtract with Carry) – odečtení dvou 8-bitových čísel a negace příznaku C (druhý operand operace odečítání je argumentem instrukce)
- CLC (Clear Carry) – nastavení příznaku C na 0 (instrukce bez explicitních argumentů)
- SEC (Set Carry) – nastavení příznaku C na 1 (instrukce bez explicitních argumentů)

Přepište následující výraz v Pascalu do ekvivalentní posloupnosti instrukcí strojového kódu procesoru 6502 (16-bitové proměnné E, F v sobě obsahují celá bezznaménková čísla a jsou uloženy na následujících adresách: E = E002h, F = F002h). Všechny operace chceme provést v celých číslech s 16-bitovou přesností bez znaménka (všechny hodnoty jsou Little Endian):

```
E := E + F + 16
```

Otázka č. 2

V kódování Unicode existuje následující přiřazení kódů jednotlivým znakům: kód 158h pro znak ‚Ř', kód 52h pro znak ‚R', kód 65h pro znak ‚e', kód 70h pro znak ‚p', kód 61h pro znak ‚a', a kód 30Ch pro kombinující znak (combining character) diakritické znaménko háček (stejné znaménko jaké je použito u znaku ‚Ř').

a) Je v kódování Unicode nějaký významový rozdíl mezi znakem 158h a posloupností znaků 52h 30Ch? Pokud ano, tak jaký?

b) Od adresy 0 chceme do paměti uložit text „Řepa" (bez uvozovek) v kódování UTF-16 ve variantě Little Endian. V šestnáctkové soustavě zapište hodnoty jednotlivých bytů paměti od adresy 0, které v sobě budou obsahovat část výše uvedeného textu v daném kódování.

Otázka č. 3

Následující hodnotu:

-256,625

zapište jako 32-bitové reálné číslo s pohyblivou desetinnou čárkou. Mantisa je 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. Výsledek zapište jako hodnoty jednotlivých bitů v pořadí zleva doprava jako MSB first.

Otázka č. 4

Mějme 8-bitový jednočipový počítač Intel 8051 s akumulátorovou architekturou a následující instrukční sadou:

- `MOV A, #`*`konstanta`* – načtení 8-bitové konstanty do akumulátoru
- `ADD A, `*`adresa`* – přičtení (bez přenosu) 8-bitové hodnoty uložené na 8-bitové adrese *adresa*

V interní paměti našeho MCU 8051 je od adresy 0 uložen následující program ve strojovém kódu 8051 (každý sudý řádek obsahuje přepis instrukce do assembleru 8051):

```
0x74 0x05
    MOV A, #5
0x25 0x03
    ADD A, 3
0x25 0x04
    ADD A, 4
```

Předpokládejte, že počítač začne tento program provádět (např. jako následek nepodmíněného skoku na adresu 0). Rozhodněte a zdůvodněte, zda můžeme určit obsah registru A po provedení všech instrukcí výše uvedeného programu, a pokud ano, tak jaká hodnota A je? **Pozor:** procesor 8051 má čistou Harvardskou architekturu.

Otázka č. 5

Předpokládejte, že máme 3 jednočipové počítače A, B, a C propojené jednou sběrnicí I^2C. Všechna připojená zařízení používají rychlost 100 kb/s, a 7 bitové adresování. Zařízení mají nastaveny následující adresy: A = 02h, B = 07h, C = A1h. Předpokládejte, že na sběrnici nikdo nevysílá, a zařízení A se rozhodne poslat zařízení B následující 2 byty: AA 0F. Nakreslete a popište průběh napětí (a jejich logických hodnot) na vodičích Serial Data (SDA) a Serial Clock (SCL) od začátku až do konce výše zmíněného přenosu (předpokládejte, že během přenosu nedojde k žádným chybám).

Otázka č. 6

Popište všechny typické stavy, ve kterých může být vlákno v běžném systému s vícevláknovým zpracováním. Popište také všechny běžné přechody mezi jednotlivými stavy vlákna a vysvětlete, v jaké situaci k danému přechodu dochází.

Otázka č. 7

Předpokládejte, že máme počítač IBM PC kompatibilní s procesorem Intel 80386DX (32-bit procesor s 32-bit datovou a 32-bitovou adresovou sběrnicí, a zvláštním 16-bit I/O adresovým prostorem). Základní deska obsahuje pouze 16-bitové ISA sloty (16 datových a 24 adresových vodičů + vodič pro rozlišení paměťové a I/O transakce [M/IO]), a neobsahuje žádný řadič velkokapacitního úložného zařízení jako je pevný disk nebo disketová mechanika.

V základní desce jsou vloženy 2 SIMM paměťové moduly s celkovou kapacitou 2 MB DRAM. V 1. ISA slotu je vložená 16-bit VGA grafická karta Trident 8900C s 512 kB dedikované video RAM. Ve 3. ISA slotu je vložena 16-bit síťová karta 3Com 3c509C, která je připojená do 10 Mbit sítě Ethernet pomocí UTP kabelu. Okolní síťová infrastruktura je vhodně nakonfigurována.

Po zapnutí počítače dojde k nabootování operačního systému MS-DOS 5.0 ze sítě (stažením jádra OS z dostupného serveru). Popište, co se v počítači děje od jeho zapnutí do začátku stahování jádra MS-DOS. Popište hlavně to, jaké instrukce (a kterých programů) CPU po celou dobu startu počítače zpracovává, a odkud je přečte.

Otázka č. 8

Předpokládejte, že chcete programovat větší množství různých aplikací, které všechny budou používat grafické uživatelské rozhraní (GUI). Každou z naprogramovaných aplikací budete chtít v binární spustitelné podobě distribuovat pro operační systémy Linux a Mac OS X. Oba tyto operační systémy mají ale rozdílné API (navzájem nekompatibilní) pro tvorbu grafického uživatelského rozhraní, a zároveň programovací jazyk, který budete používat pro programování aplikací, nemá žádnou podporu pro tvorbu GUI. Aplikace chcete programovat tak, aby byly přenositelné na úrovni zdrojového kódu mezi oběma uvedenými systémy. Jakým způsobem to nejlépe zařídíte? Jakým způsobem pak bude probíhat překlad veškerého potřebného kódu každé takové aplikace, až do stavu, kdy máme finální spustitelné soubory?

Otázka č. 9
(otázka za celkem 2 body)

Předpokládejte počítač s 32-bitovým paměťovým adresovým prostorem. V systému je nainstalována 64 kB velká paměť ROM, která je souvisle namapována na nejvyšší možné adresy v paměťovém adresovém prostoru počítače. Dále je v systému nainstalováno 0,25 GB paměti RAM, která je souvisle namapována od adresy `0` v paměťovém adresovém prostoru počítače, s výjimkou adres `1F1h` až `178h` na kterých jsou namapovány porty HDC pomocí mechanizmu MM I/O. Předpokládejte, že v Pascalu (případně v jazyce C) implementujete část firmware počítače, který bude uložený ve zmíněné paměti ROM. Napište implementaci funkce `Write` (a případně dalších procedur a funkcí, které budete potřebovat) s následujícím prototypem (viz níže), která má zařídit zapsání 1 sektoru dat (vždy `512` bytů) na pevný disk – parametr `sector` určuje číslo sektoru na disku, který se má zapsat; parametr `data` obsahuje ukazatel na `512` bytů dat v paměti, které se mají zapsat do daného sektoru (ukazatel samozřejmě obsahuje adresu 1. z `512` bytů). Funkce se vrátí ihned, jak to bude možné a nečeká na kompletní dokončení operace zápisu (tj. je asynchronní operací zápisu na pevný disk). Funkce vrací: (a) `True`, pokud došlo k úspěšnému spuštění operace zápisu, (b) `False`, pokud ještě probíhá předchozí operace zápisu na pevný disk a tedy požadovaný zápis není možné provést:

```
type
      PByte = ^Byte;
function Write(sector : Word;
               data : PByte) : Boolean;
procedure InitHarddisk;
procedure SetInterruptVector(
               intVec : Integer;
               handlerRoutine : Pointer);
```

Při inicializaci firmware počítače se mimo jiné volá i vaše procedura `InitHarddisk` (viz výše), do které můžete naimplementovat libovolnou inicializaci (např. vašich globálních proměnných), kterou budete potřebovat. Dále je vám k dispozici předpřipravená procedura `SetInterruptVector`, která do vektoru přerušení s číslem `intVec` nastaví adresu obslužné procedury předané v parametru `handlerRoutine`. Můžete očekávat, že od začátku do konce běhu obslužné procedury jsou zakázána všechna přerušení.

S řadičem pevného disku se komunikuje pomocí mechanizmu PIO. Pro iniciaci operace zápisu na pevný disk je třeba provést následující posloupnost zápisů do portů jeho řadiče (vždy je uvedena adresa, na které je daný port namapovaný):

1) `1F6h`: horních 8 bitů čísla sektoru
2) `1F4h`: spodních 8 bitů čísla sektoru
3) `1F8h`: 8-bitový identifikátor příkazu: příkaz `WRITE SECTOR` = hodnota `80h`

Poté je třeba vyčkat, až bude řadič připraven na přijímání dat – to řadič indikuje nastavením 6. bitu (číslované od 0), tzv. `BSY` (Busy), ve stavovém portu na adrese `1F8h` na hodnotu `0`. Poté je možné zapisovat jednotlivé byty (které mají být zapsány do vybraného sektoru) do datového portu na adrese `1F1h`. Zápis každého bytu do datového portu způsobí nastavení bitu `BSY` na hodnotu `1`. Před zápisem každého dalšího datového bytu je tedy třeba vždy vyčkat, až řadič opět oznámí svoji připravenost nastavením `BSY` na `0`. Každé nastavení bitu `BSY` na hodnotu `0` je řadičem disku indikováno vyvoláním přerušení číslo `14`. Pozor: v obsluze přerušení `14` je třeba ověřit, že přerušení opravdu vyvolal řadič disků a ne jiný zdroj, tj. že hodnota bitu `BSY` je opravdu `0`. Pokud přerušení pochází z jiného zdroje, je možné ho ignorovat. Předpokládejte, že není zapnutá segmentace, ani stránkování. Předpokládejte, že typ `Word` slouží pro ukládání bezznaménkových celých čísel a jeho velikost je 16-bitů.