Ciel demonstracie:

Cielom je pomocou demonstracie na realnom procesore vysvetlit nasledujuce pojmy a principy vyuzivane v CPU:

a) Startovaci kod a jeho vazbu na program main()
b) Funkciu zasobnika a jeho vyznam
c) Prerusovaci system a vektory prerusenia

Uvedene pojmy a principy su vyuzivane prakticky vo vsetkych typoch realnych CPU, ich forma a realizacia je vyrazne zavisla na cielovej CPU architekture.

Demonstracia bude realizovana s vyuzitim vyvojoveho nastroja Keil uVision (kompletne IDE pre mikrokontrolery (MCU) s jadrom 8051), ktory je predinstalovany v tzv. evaluacnej verzii vo virtualnom stroji. Podrobnejsie informacie o uvedenom IDE su dostupna na stranke:

http://www2.keil.com/mdk5/uvision/

a kompletne IDE je stiahnut na stranke (pre platformy 8051, ARM, ...):

https://www.keil.com/download/product/

Vo virtualnom stroji su predinstalovane verzie uVision pre MCU s jadrom 8051 aj ARM (pozri adresar C:\KEIL\...).
IDE pre cielove MCU obsahuje standardne vyvojove nastroje:

- asembler
- linker
- knihovnik (librarian)
- C prekladac (pre ARM platformu aj C++)
- simulator
- podporu pre ladenie v cielovej platforme (on chip debugger)
- RTOS

V dnesnej demonstracii bude dalej vyuzita len verzia pre MCU s jadrom 8051. Pre platformu 8051 je do systemu uVision este doinstalovana podpora pre vizualizaciu stavoveho automatu jadra 8051 (8051 statemachine), ktorym je riadena cinnost jadra CPU. Tento volne dostupny nastroj je mozne stiahnut zo stranky (na stranke su aj dalsie uzitocne rozsirenia):

http://www.c51.de/c51.de/Dateien/uVision2DLLs.php?Spr=EN&UIN=

Pomocou nainstalovaneho "8051 statemachine" rozsirenia je mozne v ramci demonstracie sledovat aj jednotlive fazy cinnosti jadra 8051, co moze byt zaujimave pre lepsie pochopenie napr. casovania jednotlivych instrukcii.

PRIKAD_1 - Hello World (aps_8051_hello_world.zip)

V tomto projekte ukazeme, ako je mozne prepojit MCU s "vonkajsim svetom". Na prepojenie bude vyuzita specializovana periferia - UART (Universal Asynchronous Receiver Transmitter), ktora je typicky dostupna v MCU

a vyuziva sa napr. aj na konfiguraciu sietovych komponentov (napr. Cisco rutre ...). Podrobnejsie informacie su dostupne napr. na stranke:

https://data.kemt.fei.tuke.sk/MikroprocesorovaTechnika/_web/wwwfiles/str%2006.htm

Program je na napisany v jazyku C a realizuje periodicke vypisovanie textu "Hello World" pomocou periferie UART. Pred zaciatkom vypisovania je potrebne periferiu UART nakonfigurovat, co je realizovane zapisom do SFR (specialnych funkcnych registrov), ktore su mapovane do pamatoveho priestoru MCU. Zapisom do tychto registrov je mozne definovat napr. mod periferie UART (pocet bitov, paritu, ...), rychlost prenosy a pod.

Zaroven je v priklade demonstrovany vystup logickych hodnot ("1" a "0") na vyvod procesora. Vystup je realizovany zapism pozadovanej hodnoty do registra P1 (je to opat SFR periferie, ktora je mapovana priamo na vyvody MCU portu s oznacenim P1 - pozri vyvody MCU 8051 na obrazku z minulej prednasky).

Prostredie uVison IDE umoznuje velmi detailne sledovat cinnost procesora, co je ukazane v ramci demonstracie. Podrobnejsie informacie k projektu su aj v ramci komentarov v zdrojovom kode HELLO.C

Sucastou projektu je aj tzv. STARTOVACI kod STARTUP.A51

Startovaci kod procesora je typicky (niekedy moze byt aj v C-cku ...) napisany v asembleri cieloveho procesora a spusta sa hned po resete MCU. Jeho cinnost je tak vykonavana este pred zaciatkom programu main(). Hlavnou ulohou startovacieho suboru je incializovat procesor (dolezite registre, periferie, pamat) tak, aby bolo mozne nasledne spustit programovy kod (v pripade jazyka C) programu main().

V ramci demonstracie bude ukazana cinnost vynulovania internej pamate v slucke:

```
STARTUP1:

                MOV     R0,#IDATALEN - 1
                CLR     A
IDATALOOP:      MOV     @R0,A
                DJNZ    R0,IDATALOOP
```

a tiez incializacia VRCHOLU ZASOBNIKA

```
                MOV     SP,#?STACK-1
```

Pojem ZASOBNIK a VRCHOL ZASOBNIKA je klucovy pre volanie podprogramov a cinnost prerusovacieho systemu, ktore su demonstrovane v dalsich prikladoch.

PRIKLAD_2 - Cinnost zasobnika (aps_8051_cinnost_zasobnika.zip)

Sucastou archivu ZIP je aj prehladne zobrazenie instrukcii MCU s jadrom 8051 (instrukcie aritmeticke, logicke, riadenia toku, ...). Z prehladu je zrejme, ze MCU s jadrom 8051 ma desiatky instrukcii a na prvy pohlad posobia instrukcie "kopmlikovane".
V ramci demonstracie bude naznacena filozofia instrukcneho suboru a tiez sposob zapisu v jazyku asembler (instrukcie a DIREKTIVY asemblera 8051).

INSTRUKCIA je prikaz pre PROCESOR a prekladac (asembler) ju transformuje do STROJOVEHO KODU, teda do postupnosti jednotiek a nul, ktore vie cielovy procesor interpretovat.

DIREKTIVA asemblera je prikaz PRE SAMOTNY PREKLADAC (napr. direktiva

ORG 1000h

definuje, ze prekladac ma zacat generovat instrukcie od adresy 1000 hexadecimalne, t.j. prvu instrukciu po tejto direktive umiestni na adresu 1000h v programovej pamati.)

V procesorovej technike pojem ZASOBNIK reprezentuje oblast pamate ku ktorej procesor pristupuje ako k LIFO (Last In First Out) strukture. Primarnym ucelom zasobnika je uchovavat NAVRATOVE hodnoty pri volani podprogramov (napr. CALL insrukcia) a tiez uchovavat navratove hodnoty pocas PRERUSENI (vid. dalsi priklad). Do zasobnika sa tiez ukladaju napr. automaticke premenne, ...

V ramci dmonstracie bude demonstrovana cinnost zasobnika pocas volania "podprogramu" v jazyku asembler. V prostredi Keil uVision bude mozne sledovat co sa zapisuje do zasobnika (casto sa pouziva aj slovne spojenie "na zasobnik") a ako sa takto zapisana informacia zo zasobnika vyuzije pri navrate z podprogramu.

Tiez bude ukazane, ze udaj ZAPISANY do zasobnika sa autoamticky NEMAZE!!! a v pamati procesora tak ostava urcita informacia ("STOPA"), ktoru je mozne vyuzit napr. aj k sledovaniu toho, co sa v procesore dialo. Samozrejme, tato informacia je velmi limitovana, moze sa sa vsak niekedy uspesne vyuzit pri ladeni programov a analyze chyb.

PRIKLAD_3 - Prerusenie (aps_8051_prerusenie.zip)

Sucastou procesorov su aj periferie. Periferie integrovane v procesore umoznuju "odlahcit" cinnost CPU tak, ze vykonavaju paralelne cinnosti, ktore by musela CPU realizovat "softverovo" pomocou instrukcii. Typicky priklad je pouzitie periferie CASOVACA, ktora je vyuzita v demonstrovanom priklade.

MCU nastavi periferiu casovaca tak, aby nezavisle od CPU merala pozadovany casovy interval a po jeho uplynuti upozornila CPU, ze nastaveny cas uz uplynul. CPU moze pocas tejto doby realizovat inu zmysluplnu aktivitu - napr. realizovat nejaky vypocet.

Mechanizmus, ktory umoznuje (z pohladu CPU) informovat CPU asynchronnym sposobom o nejakej udalosti (v nasom priklade o uplynuti casoveho

intervalu) je tzv. PREUSENIE procesora.

Prerusenie procesora je udalost, kedy vykonavanie "hlavneho programu" je pozastavene a CPU zacne vykonavat instrukcie z inej oblasti programovej pamate (tzv. PODPROGRAM PRERUSENIA). Oblast pamate z ktorej sa zacnu instruckie podporgramu prerusenia vykonavat sa nazyva

VEKTOR PRERUSENIA.

Obvykle ma CPU vyhradenych viacero vektorov prerusenia, ktore su napr. priradene roznym zdrojom prerusenia (napr. pre rozne casovace, UART, GPIO, ...).

Napr. klasicky procesor 8051 ma nasledujuce zdroje preruseni a vkeotorve adresy:

Prerusenie	Vektorova adresa	Cislo prerusenia
External 0	0003h	0
Timer 0	000Bh	1
External 1	0013h	2
Timer 1	001Bh	3
Serial	0023h	4

Prerusovaci system moze byt aj VIAC-UROVNOVY, co znamena, ze je mozne prerusit aj aktualne vykonavany podprogram prerusenia. Typicky sa to vyuziva na obsluhu kritickych cinnosti, kedy je potrebne obsluzit udalost z periferie, ktora ma vyssiu prioritu. Cast priorit je typicky v CPU definovana pevne, cast je uzivatelsky konfigurovatelna.

V ramci demonstracie bude ukazana cinnost prerusenia od casovaca a ukazane riesenie pomocou asemblera a tiez pomocou jazyka C. V ramci prerusenia bude mozne sledovat aj cinnost ZASOBNIKA a UKAZATELA VRCHOLU ZASOBNIKA (SP), kde bude mozne sledovat ulozenu navratovu hodnotu, kde sa riadenie procesora "vrati" po ukonceni podporgramu prerusenia. Opat sa vyuziva LIFO STRUKTURA ZASOBNIKA. Pprepnutie vykonavanych instrukcii z HLAVNEHO PROGRAMU na vykonavanie instrukcii definovanych v PODPROGRAME PRERUSENIA bude mozne sledovat aj na hodnotach PC (Program Counter).

Poznamka:

Sposob zapisu podprogramu prerusenia v jazyku C je zavisly na pouzitom prekladaci. Nie je sucastou definicie jazyka C a preto vyrobcovia prekladacov musia tuto funkcionalitu do svojho prekladaca dodat. Najcastejsie je realizovana vhodnymi DIREKTIVAMI jazyka C (co su prikazy pre samotny prekladac jazyka C) alebo pridanim novych KLUCOVYCH SLOV pre prekladac. Napr. v pripade Keil rozsireni prekladaca C51 pre platformu 8051 sa funkcia prerusenia zapisuje takto (vid. priklad projektu):

```
void timer0 (void) interrupt 1  // obsluha prerusenia casovaca T0 (vektor na
adrese 0x000B)
{
  TR0=0;                        // zastavenie casovaca T0
  TL0=PERIOD%256;               // definovanie periody (spodnych 8 bitov cisla
PERIOD)
```

```
  TH0=PERIOD/256;               // definovanie periody (hornych 8 bitov cisla
PERIOD)
  TR0=1;                        // spustenie casovaca T0
  LED = ~LED;                   // negovanie signalu pre LEDku
  pocitadlo++;                  // inkrementacia (softveroveho) pocitadla
}
```

PRIKAD_4 - Endian test (aps_8051_test_endian.zip)

Tento jednoduchy prikald umoznuje zistit, ako prekladac C51 ukalada do pamate viac-bajtove data, t.j. ci vo forme BIG ENDIAN alebo LITTLE ENDIAN (vid. predhdzajuce prednasky). V pripade 8-bitoveho procesora s jadrom 8051, ktory nema hardverovo podporovane instrukcie pre pracu s viac-bajtovymi udajmi, je to usporiadanie, ktore si zvolili tvorcovia prekladaca C51. Testovaci program vyuziva funkciu

```
int machineEndianness( void )
{
   long int i = 1;
   const char *p = (const char *) &i;
   if (p[0] == 1)                  // nizsia adresa obsahuje LSB bajt
      return LITTLE_ENDIAN;
   else
      return BIG_ENDIAN;
}
```

ktora bola preberana v ramci prednasky o usporiadani dat v pamatiach CPU.