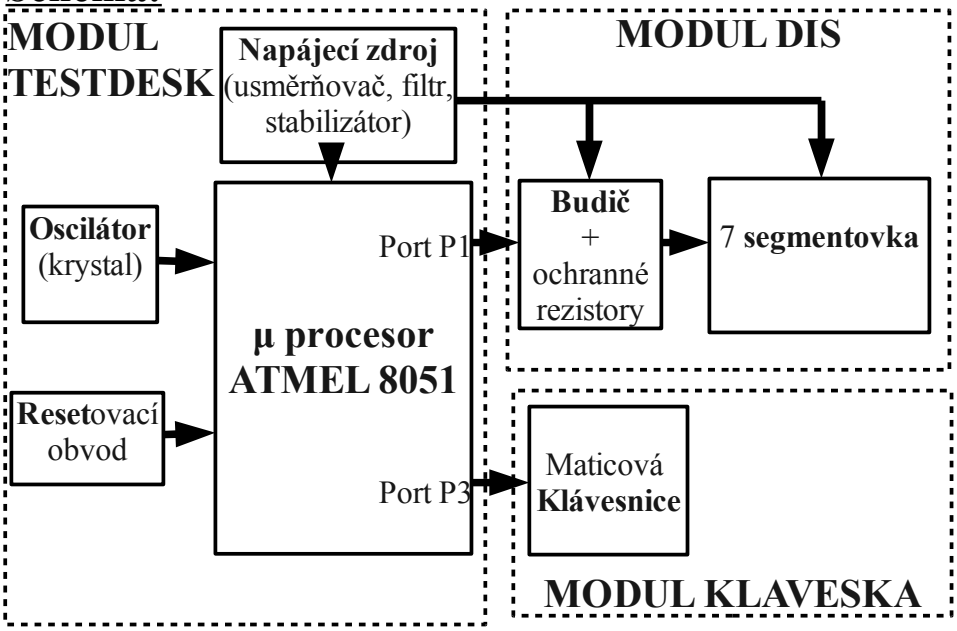
Zadání:

Při výběru svého závěrečného projektu jsem předpokládal, že výsledné zařízení bude mít nějaké praktické využití. Zvolil jsem si proto zařízení, které bude počítat aritmetický průměr jednociferných čísel (0 až 9). Tím, vznikla jednoduchá aplikace, kterou by jistě použili učitelé, kteří by pouze naťukali čísla 1-5 a program by zobrazil průměrnou známku.

Schéma:



Popis funkce:

Po startu program vyplní paměť od adresy 20H až po adresu 30H hodnotami pro sedmisegmentovku (více při popisu podprogramu writeln). Poté zobrazí „_“, což znamená, že program očekává stisk klávesy a tím nastává hlavní program.

Ten má na starost pouze dekódování stisknuté klávesy. Program aktivuje sloupec klávesnice, a vždy zapíše do registru R1 hodnotu nadcházející klávesy, a pokud je daná klávesa stisknuta, tak skočí na návěští end1, pokud není, tak tuto hodnotu přepíše další a testuje další klávesu. Jakmile proběhne celá klávesnice, jede celé testování znovu.

Při skoku na end1 program na 1 sekundu zobrazí právě stisknutou klávesu (k tomu musí z R0 přesunout číslo do R4 a vyvolat podprogram writeln), přičte její hodnotu k součtu (registr R2) a zvýší o 1 počet stisknutých kláves (registr R3)(nutné pro výpočet průměru). Opět zobrazí „_“ a vrátí se zpět do hlavního programu.

Pokud byla však stisknuta klávesa *, program nic nezapisuje do R1, ale rovnou odskočí na návěští reset. To má funkci smazání předchozí zadané číselné hodnoty, a to tak, že odečte ze součtu (registr R2) hodnotu poslední stisknuté klávesy, a dekrementuje počet (registr R3).

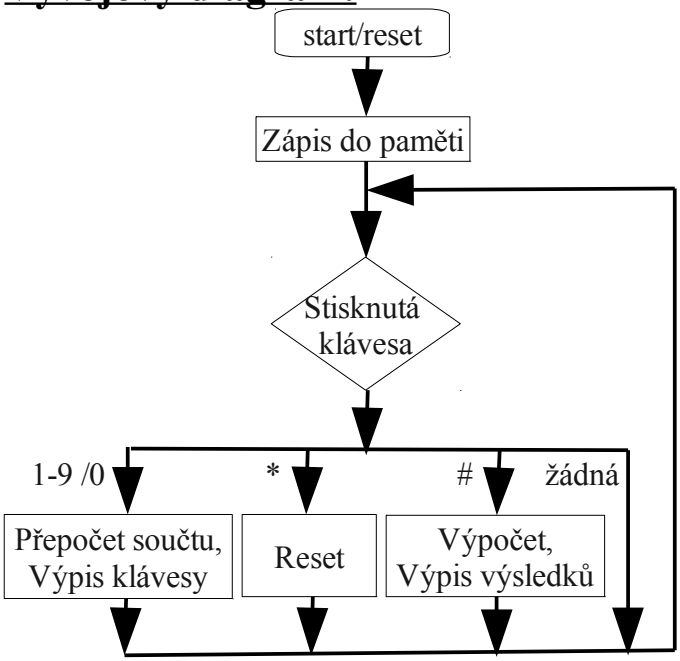
Pak zobrazí znak „|-“ , který symbolizuje umazání posledního zadaného čísla. Poté zobrazí „_“, a vrátí se zpět do hlavního programu.

Pokud byla stisknuta klávesa #, program skočí na návěští vypocet, který, jak již jeho název vypovídá provede výpočet a zobrazí výsledek. Zjednodušeně řečeno vydělí součet počtem, zobrazí =, pak celé číslo, čárku , a zbytek po celočíselném dělení vynásobí deseti, vydělí opět počtem a opět zobrazí. Toto zopakuje ještě jednou, tak tedy výsledek je zobrazen v sekundových intervalech jako znak rovná se, celá část, čárka, desetiny, setiny (zaokrouhlené dolu). Poté zobrazí „_“ a vrací se zpět do hlavního programu, kde je možné pokračovat v zadávání čísel, nebo jen znovu zobrazit výpočet a vypočtené hodnoty.

Z důvodu maximální kapacity registru součet (R2) (max. hodnota je 255) vychází, maximální počet zadaných čísel je 255 / 9 = 28 krát číslo 9, poté by program nemusel fungovat správně.

Podprogram writeln funguje tak, že číslo na výpis (jeho binární hodnotu) přesune do registru R0 (a zvýší o 20), a tím se z něj stane ukazatel do paměti. Nepřímou adresací pak najde (číslo 1 ukáže do paměti na adresu 21H atd.) jemu odpovídají vyjádření daného čísla v kódu pro sedmisegmentovku, kterou pošle na port a zobrazí.

Vývojový diagram:



Zdrojový kód:

```
;Zaverecny projekt do MIT
;\/\/\/\/\/\/\/\/\/\/\/\/\/
;Název:Vypocet aritmetickeho prumeru
;8. Martin Jašek, T3
;Datum odevzdani: 18.6.2010

;Periferie:     port P1-segmentovka
;              port P3-klavesnice

;r0-<pracovni>
;r1-<pracovni>
;r2-soucet
;r3-pocet
;r4-cislovka na vypis
;r5-stisknuta klavesa
;r6-<pracovni>
;r7-<prcovni>
        mov 20H,#11000000B
        mov 21H,#11111001B
        mov 22H,#10100010B
        mov 23H,#10110000B
        mov 24H,#10011001B
        mov 25H,#10010010B
        mov 26H,#10000011B
        mov 27H,#11111000B
        mov 28H,#10000000B
```

```
        mov 29H,#10011000B

        mov 30H,#11110111B      ;volba (znak "_")

        mov r4,#10H
        call writeln

zac:    mov p3,#11011111B
         call   manip
           mov r1,#1
           jnb p3.3,end1
           mov r1,#4
           jnb p3.2,end1
           mov r1,#7
           jnb p3.1,end1
           jnb p3.0,reset

        mov p3,#10111111B
         call   manip
           mov r1,#2
           jnb p3.3,end1
           mov r1,#5
           jnb p3.2,end1
           mov r1,#8
           jnb p3.1,end1
           mov r1,#0
           jnb p3.0,end1

        mov p3,#01111111B
         call   manip
           mov r1,#3
           jnb p3.3,end1
           mov r1,#6
           jnb p3.2,end1
           mov r1,#9
           jnb p3.1,end1
           jnb p3.0,vypocet

        jmp zac

end1:   mov a,r1        ;zatim je stisknuta klavesa v r1
        mov r4,a        ;stisknuta klavesa bude vypsana
        mov r5,a        ;prave stisknuta klavesa

        inc r3          ;pocet

        add a,r2        ;zvyseni souctu o prave stisknutou klavesu
        mov r2,a        ;soucet je v r2
        call writeln
        call delay
        mov r4,#10H
        call writeln
        jmp zac

reset:   dec r3
         mov a,r2
         subb a,r5
         mov r2,a
         mov p1,#10001111B
         call delay
```

```
        mov r4,#10H;znak pro vyckani dalsi klavesy
        call writeln
       jmp zac

vypocet: mov a,r2    ;presun souctu
        mov b,r3     ;presun        poctu
        div ab
        mov r4,a     ;cislovka na vypis je v r4
        mov p1,#10110111B       ;znak "="
        call delay
        call writeln
        call delay
        mov p1,#11111011B       ;pseudo desetinna carka
        call delay

        mov r1,#2
for:    mov a,#10
        mul ab       ;zbytek zveci 10x
        mov a,b      ;max. 10x25 neni vetsi nez 255-vysledek je v b
        mov b,r3     ;pocet
        div ab
        mov r4,a     ;cislovka na vypis je v r4
        call writeln
        call delay

        djnz r1,for
        mov r4,#10H;znak pro vyckani dalsi klavesy
        call writeln
       jmp zac

writeln: mov a,r4     ;cislovka na vypis je v r4
        add a,#20H
        mov r0,a
        mov p1,@r0
       ret

delay:  mov r6,#79
mili2:   mov r7,#255;zpozdeni cca 1 sec
mili1:    call manip
        djnz r7,mili1
       djnz r6,mili2
       ret

manip:mov r0,#16   ;zpozdeni 50us
mikro: nop
       djnz r0,mikro
       ret
end                ;konec. dekuji za pozornost
```

Návrh podprogramů pro časové zpoždění:

Při návrhu jsem uvažoval: Krystal 12MHz, tedy 1SC=1µs. n je počet opakování cyklu:
podprogram manip: Zpoždění 50 mikro sekund: n=50/(3*1)=16x

```
manip: mov r0,#16     ;zpozdeni 50us
mikro: nop
       djnz r0,mikro
       ret
```

podprogram delay: Zpoždění 1 sekunda: n=1000 000/(3*50)=6 667x → je tedy potřeba použít další smyčku: zpoždění 50µs proběhne 255, tím získáme zpoždění t=(50*255)+3=12,7ms a to zopakujeme n=1000/12,7=79x:

```
delay:  mov r6,#79
mili2:   mov r7,#255  ;zpozdeni cca 1 sec
mili1:    call manip
          djnz r7,mili1
         djnz r6,mili2
         ret
```

Závěr:

Jak již jsem popsal v úvodu (Zadání), toto je typická aplikace pro mikroprocesory, obzvláště pak řady 8051. Zařízení funguje bez problémů a hezky se zde ukazuje ten fakt, že i složitější výpočty jako desetinné dělení lze zapsat pomocí assembleru(na čemž stojí všechny dnešní počítače).