# Popis problému

Na vstupu bude zadán dělenec a dělitel (unit32). Program vypočítá a zobrazí podíl a zbytek (unit32) metodou se zpětným přičítáním.

# Řešení problému

Nejprve je potřeba získat vstupy od uživatele, které je třeba následně uchovat v paměti pro jejich další zpracování metodou se zpětným přičítáním. Ke vstupům bude vhodné přidat výstupy, které budou určovat, v jakém rozsahu se má uživatel pohybovat. Princip metody zpětného přičítání spočívá v postupném odčítání dělitele od dělence (zpětné přičítání), dokud výsledek odčítání není menší než dělitel. Tento výsledek se poté prohlásí za zbytek. Navíc v každém kroku (průběhu odčítání) se inkrementuje podíl (na začátku inicializovaný na 0) o 1.  Zbytek se pak bude nacházet v registru ER5 a podíl v registru ER6.

# Popis programu

## *Datová sekce*

Začíná vyhrazením 100 bytů místa v paměti příslušným vstupům (proměným: VSTUP_Delenec a VSTUP_Delitel). Dále jsou definovány proměnné výstupních řetězců pro navedení uživatele. Parametrické bloky proměnných se zarovnáním adres pro parametrické bloky align 2 (adresy musí být dělitelné 4). V závěru datové sekce se připravuje zásobník (zarovnání align 1 „sudé adresy“ a vymezení místa v paměti).

## *Kódová sekce*

Za prvé se načte výstup a vstup dělence, k čemuž se využívají kódy služeb PUTS (0x114 → R0), GETS (0x113, R0), SYSCALL (0x1FF00 → pod touto adresou se v paměti nachází podprogram pro zpracování). Následně jsou vstupy převedeny programem z ascii kódu na hexadecimální reprezentaci, podprogramy Pocet_mist, Reset, Decimal_to_hex, Hexadecimal, Nasobeni, Vysledna_Hex. Po převední se v ER3 nachází dělenec a v ER4 dělitel. Podprogramy Vypocet, Vysledek, Metoda_Pricitani vypočítají podíl → registr ER6 a zbytek → registr ER5. Na závěr podprogramy Hex_to_decimal, Vypocet, Okamzity_prevod, Posun, Posun_podilu, Zapis_podil, Navyseni_radu převedou hexadecimalní reprezentaci čísel zpět na unit32 a skočí se na návěští KONEC, která probíhá v nekonečné smyčce .

## Souhrn podprogramů a návěští

- Slouží k zevrubnému představení, jak podprogram funguje, jelikož jeho název nemusí vždy vystihovat jeho funkci.

**Pocet_mist:** → Na základě ascii podoby čísla spočte počet míst. Díky @ER2 uloží do R0L 8 bitové slovo a pak testuje jestli se rovná 0x0A. Ne, inkerementuje se ER2 a ER3 „počet míst“ a opakuje se cyklus. Ano skok na návěští Reset.

**Reset:** → Připraví regsitry na skok do Decimal_to_Hex.

**Decimal_to_hex:** → Probíhá ve smyčce, dokud nenačte celé číslo. Ascii cifru převede na decimální, inkrementuje ER2 a skočí do Hexadecimal.

**Hexadecimal:** → Slouží k určení, v jakém řádu se nachází cifra tím, že dekrementuje ER5, dokud není roven 0 (z ER3 se v předchozím podprogramu přesunul počet míst do ER5). Při každé dekrementaci skočí do Nasobeni.

**Nasobeni:** → Simuluje nasobeni #0x0A a decimální cifry (nahrazení instrukce mulxu.w, která nebyla dostačující pro 32 bitové registry).

**Vysledna_Hex:** → Sníží počet míst vzhledem k další cifře (dekrementace ER3). Do registru ER4 přičte velikost předchozí cifry, čímž nakonec dojdeme k výslednému převodu.

**Vypocet:** → Větví program vzhledem k vztahu mezi dělencem a dělitelem. Dělenec je menší, tak skok na návěští Vysledek, jinak skok do Metoda_Pricitani.

**Vysledek:** → Podíl nastaví na 0 a zbytek na dělenec.

**Metoda_Pricitani:** → Reprezentuje průběh již výše zmíněné metody.

**Hex_to_decimal:** → Pokud převáděné číslo je menší než deset, tak skočí na návěští Okamzity_prevod. Pokud ne, skočí do Vypocet, kde zjistí podíl a zbytek. Následně skočí do Posun. Po návratu z Posun logickým součtem registru ER1 a ER5 (zbytek) umístí cifru na správné místo a pokračuje ve smyčce, dokud zbytek není menší než 10, pak skočí na návěští Zapis_podil.

**Posun:** → Z registru R2L zjistí, jakého je převáděná cifra řádu, a podle toho jí umístí v ER5, pak skočí na návěští Navyseni_radu.

**Navyseni_radu:** → Inkrementuje řád (R2L) pro další cifru.

**Zapis_podil:** → Skočí do Posun_podilu a následně udělá logický součet ER6 (podíl) a ER1 (ostatní cifry čísla).

**Posun_podilu:** → Principem je stejný jako Posun, ale pracuje s registrem ER6 na rozdíl od Posun (pracoval s ER5).

**Okamzity_prevod:** → Rovnou zapíše převáděnou cifru logickým součtem.

**Test_vstup:** → Pomocí 3 testovacích podmínek zkontroluje, jestli uživatel zadal čísla 0-9. Pokud uživatel nezadá číslo, pak se v programu skočí na návěští EROR, které vypíše hlášku a vrátí se zpět na úplný začátek programu.

## Tabulka

| Proměnná | Umístění v paměti |
|---|---|
| VSTUP_Delenec | FF4000 |
| VSTUP_Delitel | FF4064 |
| VYSTUP_Delenec | FF40C8 |
| VYSTUP_Delitel | FF4103 |
| VYSTUP_Eror | FF413E |
| stck (zásobník) | FF41E0 |

| Podprogram | Obsah Stack Pointru při skoku do podprogramu (ER7) |
|---|---|
| Pocet_mist | 00FF41DC |
| Test_vstupu | 00FF41D8 |
| Decimal_to_hex | 00FF41D8 |
| Hexadecimal | 00FF41D4 |
| Nasobeni | 00FF41D0 |
| Vypocet | 00FF41DC |
| Metoda_Pricitani | 00FF41D8 |
| Hex_to_decimal | 00FF41DC |
| Vypocet | 00FF41D8 |
| Posun | 00FF41D8 |
| Posun_podilu | 00FF41D8 |

## Závěr

Program jsem se snažil otestovat pro různé typy vstupů. Zahrnul jsem i situace, kdy dělitel je větší, menší, pořípadě stejný jako dělenec. Pro všechny vstupy, co jsem zadával, poskytoval validní výsledky. Problém však nastává, pokud je jako dělenec zadáno velmi vyskoé číslo. Například to úplně nejvyšší možné (4 294 967 295). Naopak jako dělitel velmi malé - například 2. Pak je třeba počítat s tím, že program neposkytne validní výsledek, jelikož podíl, převedený z hexadecimální do decimální reprezentace, se nevejde do 32 bitového registru.