Feladat:

Belső memória bitindexének kiszámítása, azaz annak kiszámítása, hogy az IRAM 0..255 című cellájának adott sorszámú bitje (0..7) hányadik bit a memória kezdetétől számítva (tehát az eredmény 0..2047 közé esik). Bemenet: a cella címe, az adott bit sorszáma 1-1 regiszterben. Kimenet: a bitindex (2 regiszterben).

Problémaleírás:

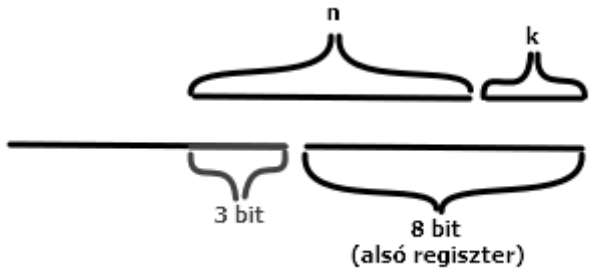
Tekintve, hogy a bitindex értéke 8-cal nő vagy csökken, ha a cellasorszám értékét eggyel növeljük/csökkentjük (0 alá, 255 fölé nem megyünk), és hogy egyesével változtatva bitszámot a bitindex is egyesével változik, a bitindex értéke megegyezik ezzel az értékkel: $8 * cellasorszám + bitszám$.

A cellasorszám nyolcszorosának előállítását tanultak szerint egyszerűen elő lehet állítani, ugyanis a szám végére ha írunk még három darab 0-t, akkor pont a cellasorszám nyolcszorosát kapjuk meg, 11 biten. Ehhez hozzá kell még adnunk a bitszámot. Mivel az egy 3 bites kifejezés, így csak az utolsó 3 bit változhat, ami ezelőtt 000 volt. Ez pont kapóra jön nekünk, hiszen így biztosak lehetünk abban, hogy az eredmény felső 8 bitje a cellasorszám értéke, alsó 3 bitje pedig a bitszám értéke.

A problémát tehát így írhatjuk le:

Bemeneten kapunk egy 8-bites számot és egy 3 bites számot, kimenet legyen ez: egy 8 bites regiszter, melynek felső 5 bitje 0, alsó 3 bitje a 8 bites szám 3 legmagasabb bitje, és egy másik 8 bites regiszter, melynek felső 5 bitje a 8 bites szám alsó 5 bitje, alsó 3 bitje pedig a 3 bites bemenet.

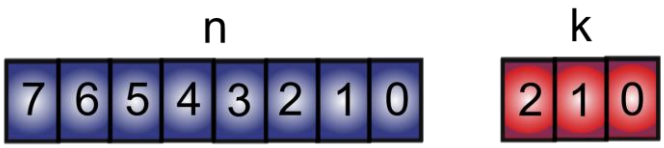
Az ábrán n-nel jelölve a cellasorszámot, k-val a bitszámot, a szerkezet látható.



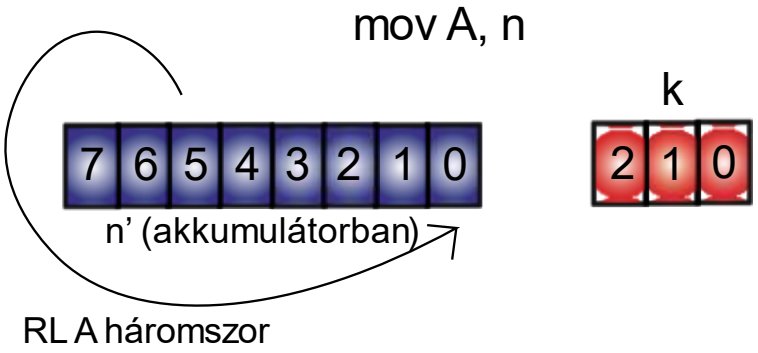
Használt algoritmus szubrutinként:

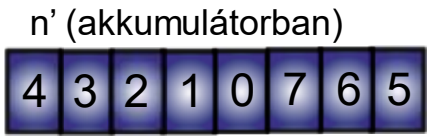
Az algoritmus egyszerű, de hatékony. A cellasorszámot (innentől legyen n) 3 bittel balra shifteljük (carry nélkül). A felső kimeneti regiszterbe beletesszük n és 00000111 bitenkénti AND eredményét, az alsóba beletesszük 11111000 és n bitenkénti AND eredményét, majd ehhez hozzáadjuk a bitszámot (kikommentelve a kódrészlet ami biztonság kedvéért 00000111-gyel bitenként ÉS-elné a bitszámot).

Vizuális reprezentálás:

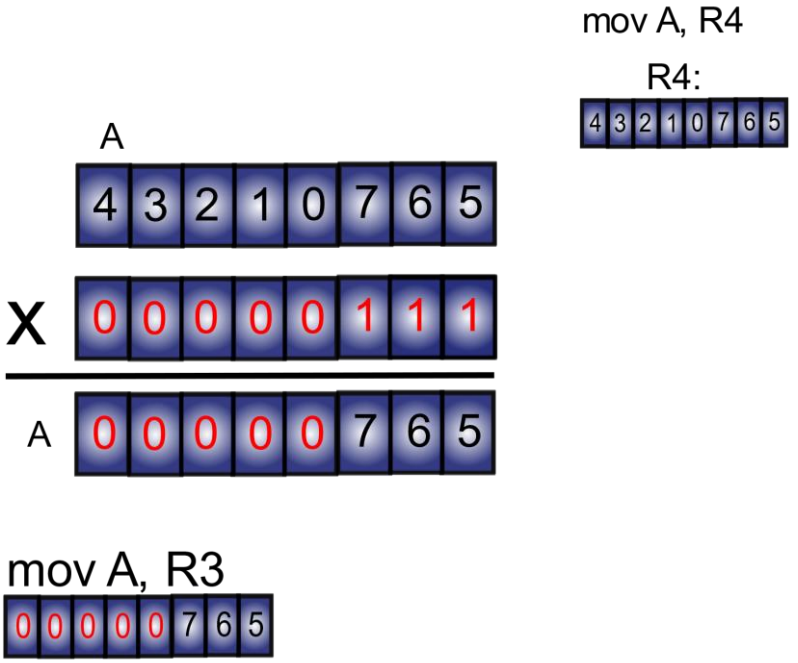


A bemenetek, n jelöli a cellaszámot, k a bitszámot. A számok a cellákban reprezentatívak, azt mutatják, hogy a kezdetben 0-7 számozású bitek közül minden utasítás hol van.

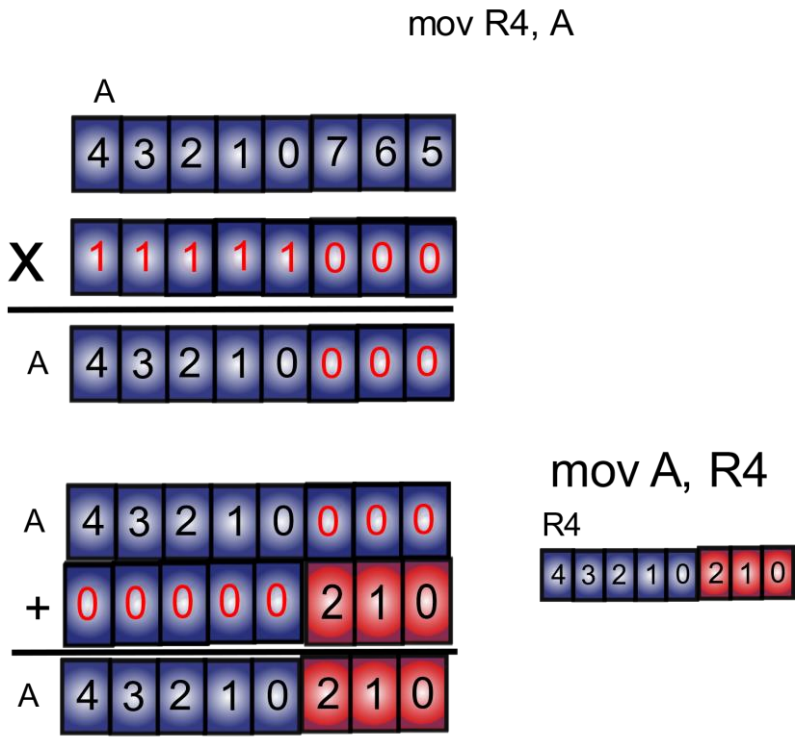




Bemásoljuk az akkumulátorba n-t, majd 3-szor balra forgatjuk.



Az akkumulátort kimásoljuk R4-be. Piros számjegyek konkrét bitértekeket jelentenek. Az akkumulátort éselve bináris 00000111-gyel, az akkumulátor alsó 3 bitjén csak n felső 3 bitje marad, ezt kimásoljuk R3-ba, ez lesz a bitindex két regiszterének felső kimeneti regisztere.

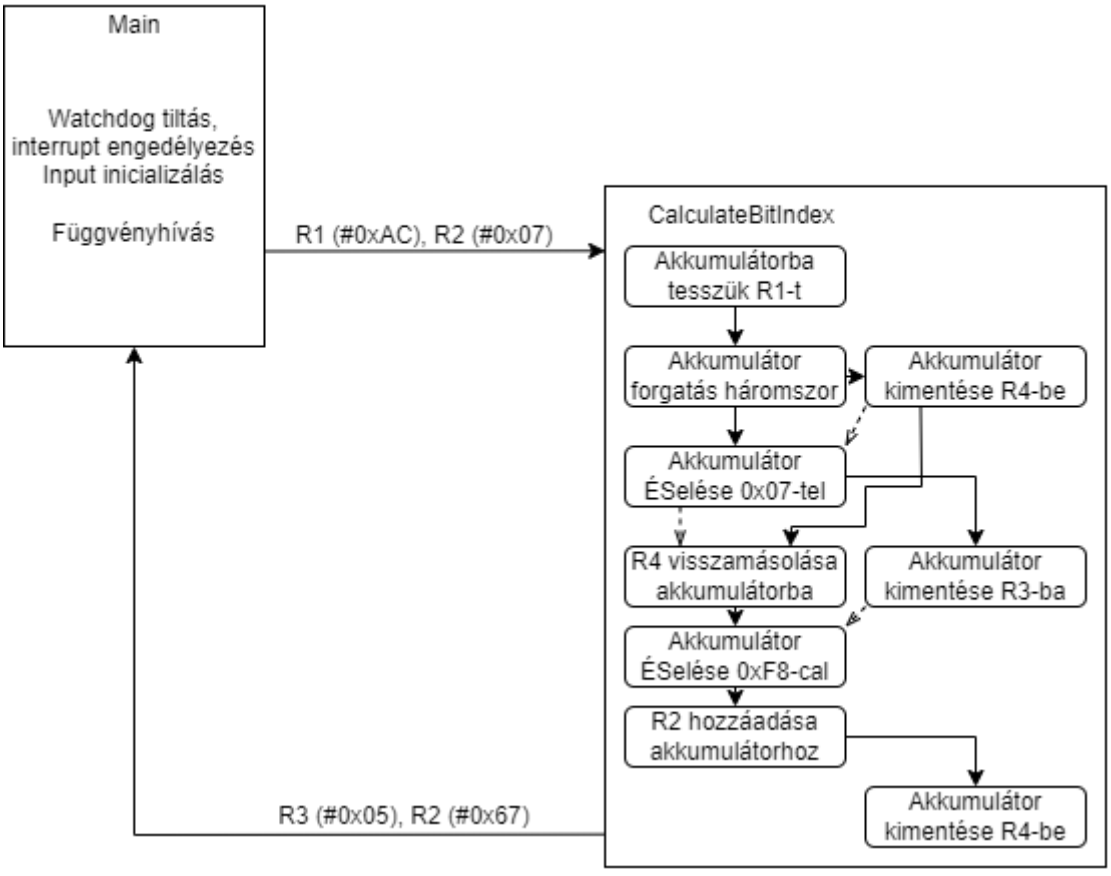


Az R4 regiszter értékét visszamásoljuk az akkumulátorba, azt összeéseljük bináris 11111000-val, az akkumulátor alsó 3 bitje 0, felső 5 bitje az a cella alsó 5 bitje. Ehhez hozzáadjuk a bitszámot, s a kapott eredményt bevisszük az R4 alsó kimeneti regiszterbe.

A kimenet:

Folyamatábra:

(Szaggatott nyíl: amely utasításra mutat a nyíl, csak akkor kezdhetődhet meg, ha az utasítás melyből a nyíl indul már lefutott, ez a kód jelentőségéből azt jelöli, hogy hamarabb szerepel a kódban)



Felhasznált forrás:

[1] BME-VIK: Mikrokontroller alapú rendszerek c. tárgy előadásjegyzetek.

[2] Keil: 8051 Instruction set
8051 Instruction Set Manual: Instructions (keil.com)

[3] Assembly bináris, decimális, hexa szám, string szintaxis

NASM - The Netwide Assembler

Használt szoftverek, hardver, eszközök:

-Simplicity Studio™ Simplicity Eclipse-based IDE v4: a program megírásához, lefordításához, szerkesztéséhez.
-EFM8 Busy Bee 3 Starter Kit MCU (gyártó: Silicon Labs): a program lefuttatásához, ellenőrzéséhez.
-Microsoft Word 2019 a dokumentáció megírásához.
-Adobe Illustrator 2021 ábrakészítéshez.
-PicPick ábraszerkesztéshez.