

Diagrama de Flux: Proiect ASM

Analiza Arhitecturală și Logică a Programului pe 16 Biți (8086
Assembly)

Viziune de Ansamblu

Procesul complet de la citirea octetilor hexazecimali până la generarea tabelului final rotit.

Fluxul Principal de Execuție



Modulul 1: Citirea și Validarea



Buffered Input

Se utilizează funcția DOS 0Ah. Primele 2 locații din buffer rețin mărimea maximă și mărimea actuală a sirului citit.



Filtrare Spatii

Algoritmul ignoră caracterele ' ' (20h) pentru a permite introducerea octeților cu spațiu între ei.



Verificare Limitări

Dacă $nr_el < 8$ sau $nr_el > 16$, se afișează msg_eroare și programul revine la START.

Modulul 2: Calculul cuvântului C

Algoritm Bitwise

Cuvântul C este format din 16 biți (DW) calculați prin trei metode diferite pentru a asigura variația datelor.

$$C_{8-15} = \left(\sum_{i=0}^n S_i \right) \bmod 256$$

Detaliere Structură

- ✓ Biții 0-3: XOR între nibble superior ($S[0]$) și nibble inferior ($S[n]$).
- ✓ Biții 4-7: OR cumulat al bitilor 2,3,4,5 din toți octetii.
- ✓ Biții 8-15: Suma aritmetică a întregului sir.

Reprezentarea Registrului C

16
Biți Total

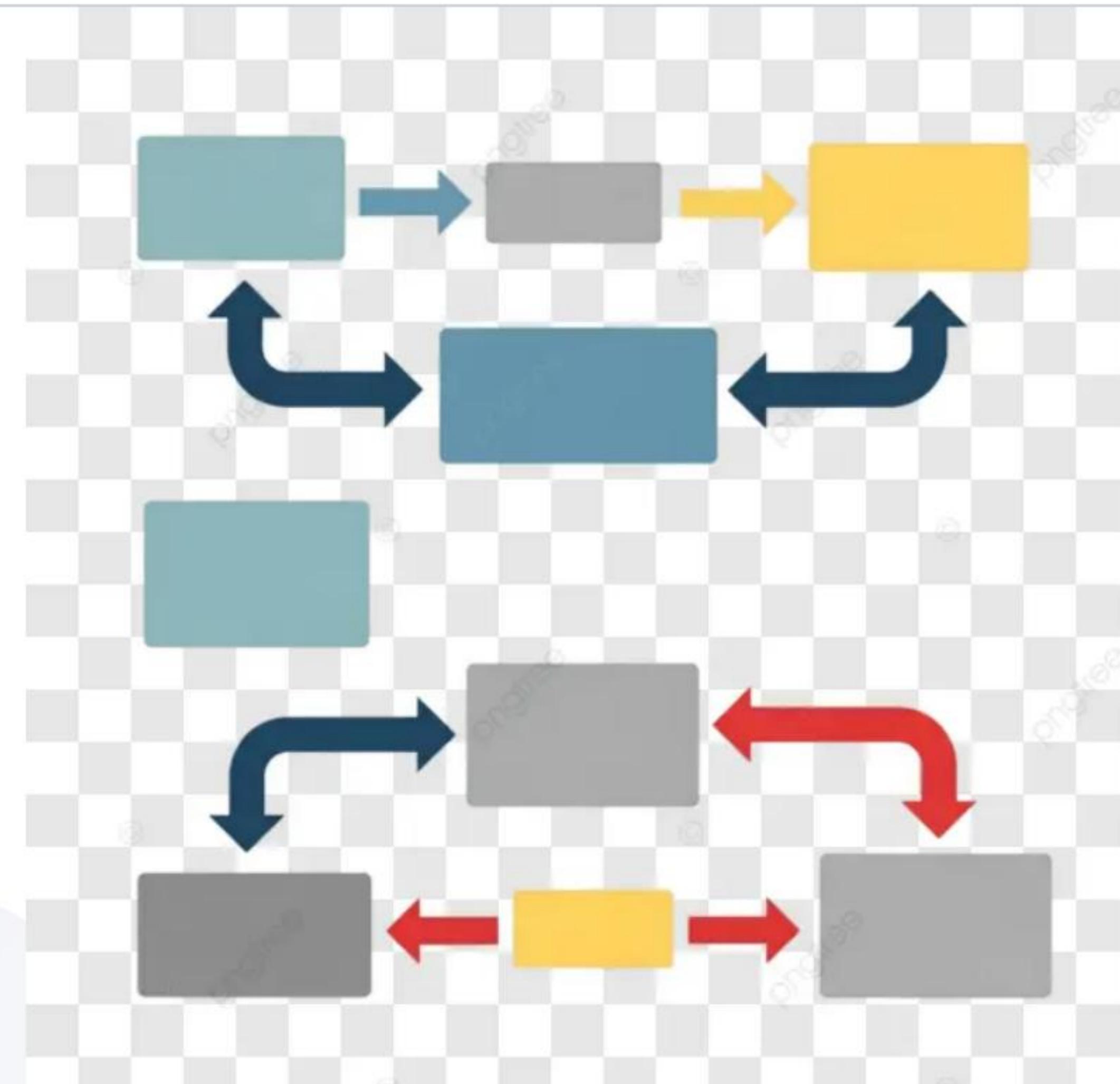
Validarea În Timp Real

Rezultatul este afișat imediat după conversie în format **HEX** și **BINAR** prin apelarea subrutinelor dedicate.

Exemplu calcul biți 4-7:

```
SHR AL, 2 / AND AL, 0Fh / OR DL, AL
```

Modulul 3: Sortare și Analiză



Bubble Sort (Descrescător)

- ⬇️ **Bucla Externă (CX):** Rulează de $(nr_el - 1)$ ori.
- ⌚ **Bucla Internă (SI):** Compară [SI] cu [SI+1] folosind instrucțiunea CMP AL, BL urmată de JAE (Jump if Above or Equal).
- ⇄ **Swap:** Se execută doar dacă elementul curent este mai mic decât următorul.

Căutarea Maximului de Biți '1'

Algoritmul de Numărare

Pentru fiecare octet, se execută o buclă de 8 iterații:

- 1. SHL AL, 1:** Mută cel mai semnificativ bit în Carry Flag.
- 2. ADC BL, 0:** Adună valoarea Carry (0 sau 1) la contorul BL.
- 3. Verificare Minim:** Dacă $BL < 4$, elementul este ignorat (conform cerinței).
- 4. Update Maxim:** Se salvează poziția ($SI + 1$) în pos_max.

 Bit counting visualization

Modulul 4: Transformarea Finală

Operație	Descriere Logică	Instrucțiune ASM
Calcul N	Suma primilor doi biți (bit 7 și bit 6).	ROL + ADC
Rotire	Rotire circulară la stânga cu N poziții.	ROL AL, CL
Afișare B	Prefixare cu 'B:' și afișare binară.	CALL PRNT_BIN
Afișare H	Prefixare cu 'H:' și afișare hexazecimală.	CALL PRNT_HEX

$$N = Bit_7 + Bit_6 , \quad S_{final} = ROL_N (S_{sortat})$$

Subrutine și Utilitare



CONV_CHAR

Convertor ASCII ('0'-F') în numeric (0-15). Gestioneză atât cifrele cât și literele (ajustare cu 30h/37h).



PRNT_BIN

Afișează registrul AL bit cu bit folosind rotiri și funcția DOS 02h (Character Output).



PRNT_HEX

Descompune AL în două nibbles, convertindu-le în ASCII pentru afișarea hexazecimală.

Final de Execuție

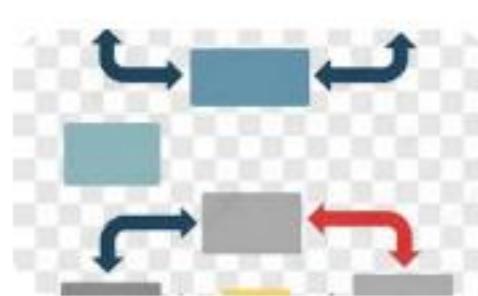
Programul se încheie cu instrucțiunea MOV AH, 4Ch / INT 21h, returnând controlul către sistemul de operare.

Întrebări?

Explicații suplimentare despre logică, registre sau fluxul de date.

Prezentare Generată pentru Analiza Proiect ASM

Image Sources



https://png.pngtree.com/png-vector/20260101/ourlarge/pngtree-abstract-diagram-illustrating-interconnected-processes-and-data-flow-with-directional-arrows-png-image_18387562.webp

Source: pngtree.com



<https://en.pimg.jp/058/954/186/1/58954186.jpg>

Source: www.pixtastock.com