Univerzitet u Beogradu - Elektrotehnički fakultet

Katedra za elektroniku

Huffman Coding

Implementacija u MASM asemblerskom jeziku

13E043RE - Računarska elektronika

Radomir Vranjevac 2017/0103
Dušan Ilić 2017/0070

1. jul 2020.

Sadržaj

Zadatak

Potrebno je napisati program u **MASM** jeziku koji vrši *Huffman*-ovo kodovanje simbola. Ovo kodovanje zasnovano je na verovatnoći pojavljivanja simbola gde se simboli sa najvećom verovatnoćom pojavljivanja koduju sa kodnom rečju najmanje dužine.

Nakon pokretanja programa od korisnika se traži da unese naziv tekstualne datoteke nad kojom želi da primeni mehanizam *Huffman*-ovog kodovanja. Kao rezultat kodovanja na konzoli se ispisuje učestanost pojavljivanja pojedinih simbola kao i kodne reči koje su dodeljene pojedinom simbolu.

Uvod

Huffman-ovo kodovanje predstavlja algoritam za kodovanje simbola, gde dužina koda svakog simbola zavisi od učestanosti ponavljanja datog simbola u određenom tekstu, odnosno tekstualnoj datoteci. Cilj je da se kodovanjem smanji potreban memorijski prostor za čuvanje sadržaja nekog fajla.

Huffman-ovo kodovanje spada u *prefix* kodove, odnosno zadovoljava pravilo da nijedan kôd nije prefiks nekog drugog koda (kodovi u kojima je svaki simbol predstavljen istim brojem bita automatski zadovoljavaju pravilo prefiksa). Ovo predstavlja dobru osobinu jer znatno olakšava dekodovanje.

U daljem tekstu će detaljno biti objašnjen postupak kodovanja.

Struktura projekta

Prilikom izrade projekta, projekat je podeljen na logičke celine. Ceo projekat se sastoji iz sledećih delova i fajlova:

1. **Konfiguracija** - sadrži fajl u kojem se nalaze definicije konstanti i parametara korišćenih u projektu, poput maksimalne dužine niza itd.
   - `configuration.inc`
2. **Složene strukture** - sadrži fajl u kojem se nalaze definicije složenih struktura podataka koje su korišćene prilikom izrade projekta.
   - `structures.inc`
3. **Rad sa tekstualnom datotekom** - sadrži fajlove za deklaracije i definicije procedura korišćenih za obradu sadržaja tekstualne datoteke.
   - `file_process.inc`
   - `file_process.asm`
4. **Formiranje *Huffman*-ovog stabla i rad sa njim** - sadrži deklaracije i definicije procedura korišćenih za rad sa *Huffman*-ovim stablom.
   - `huffman_tree.inc`
   - `huffman_tree.asm`
5. **Glavni program** - sadrži glavni program koji koristi sve pomoćne procedure definisane u ostalim fajlovima.
   - `main.asm`

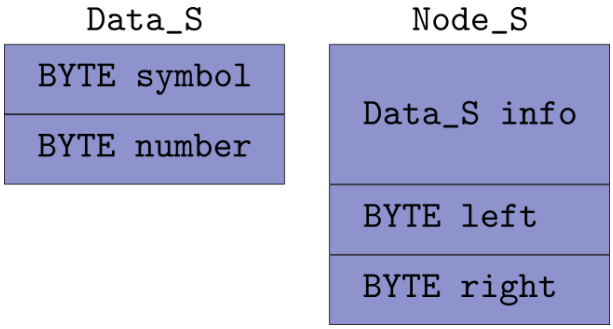
Include fajlovi

Svaki `.inc` fajl zaštićen je od višestrukog uključivanja u projekat definisanjem makroa, čije se postojanje proverava prilikom uključivanja u projekat. Primer koda koji ovo obezbeđuje prikazan je u nastavku.

```
ifndef _FILENAME_INC_
_FILENAME_INC_ = 0

; content goes here

endif
```

Strukture podataka

Grafički prikaz struktura koje su korišćene prikazan je na slici 1.



Slika 1: Strukture podataka

Polje `symbol` strukture `Data_S` sadrži karakter, a polje `number` sadrži njegov broj ponavljanja. Kao posledica odabrane veličine podatka za polje `number`, nameće se ograničenje od 255 pojavljivanja karaktera (ali i ukupnog broja karaktera jer se ista veličina koristi u `Node_S`).

To ograničenje bi moglo na relativno jednostavan način biti prevaziđeno; ali bi dovelo do strukture koja ne može da se učita u registar (koristeći 386 arhitekturu), što je korišćeno u dostavljenoj realizaciji. Još jedno od unapređenja programa bilo bi dodavanje podrške za karaktere van `ascii` koda, ili čak kodove sa promenljivom veličinom karaktera (kakav je `UTF-8`).

Struktura `Node_S` predstavlja čvor u *Huffman*-ovom stablu . Polja `left` i `right` koriste se za određivanje deteta datog čvora. *Huffman*-ovo stablo se čuva u obliku dvostruko ulančanog niza, gde polja `left` i `right` sadrže indekse čvorova oba deteta u istom nizu.

Rad sa datotekom

Deklaracaije procedura za rad sa datotekom su nalaze u fajlu `file_process.inc`. U njemu su navedene deklaracije 5 procedura koje će detaljnije biti opisane u nastavku. Definije svih procedura nalaze se u fajlu `file_process.asm`.

ReadText

`ReadText` - kao parametre dobija početnu adresu niza na kojoj se nalazi ime datoteke koju treba pročitati, kao i početnu adresu niza u kojem će biti smešten sadržaj datoteke.

U navedenoj proceduri koristili smo procedure iz `irvine32.lib` biblioteke, i to:

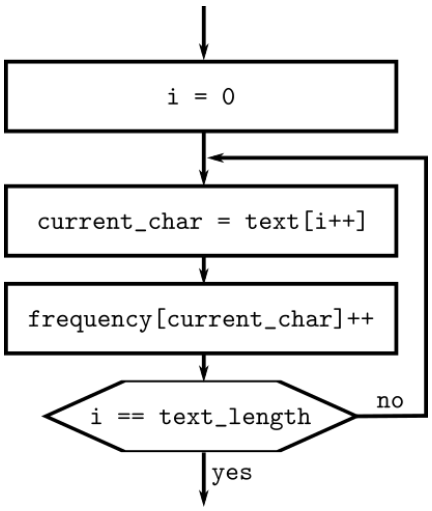
- `OpenInputFile` - za otvaranje datoteke
- `ReadFromFile` - za čitanje datoteke
- `CloseFile` - za zatvaranje datoteke
- `WriteString` - za ispisivanje eventualnih grešaka prilikom izvršavanja

Na kraju procedure se u registru `eax` smešta broj pročitanih karaktera.

Count

`Count` - kao parametre dobija početnu adresu niza na kojoj se nalazi tekstualni sadržaj, dužinu tekstualnog sadržaja, kao i početnu adresu pomoćnog niza koji će po izvršavanju procedure sadržati broj ponavljanja svih `ascii` karaktera u tekstualnom sadržaju.

Osnovna ideja je bila koristiti `ascii` kôd svakog karaktera kao indeks niza, i inkrementirati element niza prilikom svakog nailaska na određeni karakter. Vremenska složenost ovog algoritma je minimalna, a ne zahteva previše statički alocirane memorije. Algoritam je prikazan na slici 2.



Slika 2: Algoritam za određivanje broja ponavljanja karaktera

Trim

`Trim` - kao parametre dobija početnu adresu niza koji sadrži broj ponavljanja svih `ascii` karaktera u tekstualnom sadržaju, kao i početnu adresu niza koji će po izvršavanju procedure sadržati informaciju o broju ponavljanja karaktera koji se javljaju bar jednom.

Rezultujući niz je niz strukture `Data_S` koja sadrži informaciju o karakteru i njegovom broju ponavljanja. Sve što ova procedura radi je da prolazi kroz ulazni niz, i kada naiđe na karakter koji se javlja bar jednom, pravi strukturu i popunjava je odgovarajućim podacima, a zatim dodaje novi element na kraj niza. Takođe, broj takvih karaktera se na kraju procedure čuva u registru `eax`.

Sort

`Sort` - kao parametre dobija početnu adresu niza tipa `Data_S` koji sadrži informacije o broju ponavljanja svakog karaktera koji se javlja u tekstualnom sadržaju, kao i dužinu datog niza.

Rezultat izvršavanja procedure je neopadajuće sortiran niz, sortiran po broju ponavljanja datog karaktera. Sortiranje je rađeno `bubble sort` algoritmom.

PrintResults

`PrintResults` - kao parametre dobija početnu adresu niza tipa `Data_S` koji sadrži informacije o broju ponavljanja svakog karaktera koji se javlja u tekstualnom sadržaju, kao i dužinu datog niza.

Ova procedura nije od značaja za samo funkcionisanje programa, već služi za proveru dobijenih rezultata prilikom testiranja. Rezultat izvršavanja je ispis svkaog karaktera i njegovog broja ponavljanja na konzolu.

U navedenoj proceduri koristili smo procedure iz `irvine32.lib` biblioteke, i to:

- `WriteString` - za ispisivanje niza karaktera
- `WriteChar` - za ispisivanje jednog karaktera
- `WriteDec` - za ispisivanje broja

Formiranje stabla

Nakon sortiranja struktura `Data_S` u nerastućem poretku po broju pojavljivanja karaktera, potrebno je formirati *Huffman*-ovo stablo.
Osnovni princip iza *Huffman*-ovog algoritma je združivanje dva čvora sa najmanjim brojem pojavljivanja. Ti čvorovi u prvoj iteraciji predstavljaju listove stabla, odnosno dva najređe korišćena karaktera u ulaznoj datoteci. Od njih se pravi novi čvor, sa ekvivalentnim brojem pojavljivanja jednakim zbiru pojavljivanja dva karaktera od kojih je sačinjen.

Primera radi, biće opisan korak algoritma formiranja stabla na ulaznim podacima sledećeg sadržaja (navedeni su karakter i broj njegovog pojavljivanja):

c	z	k	m	c	u	d	l	e
n	2	7	24	32	37	42	42	120

Počinje se združivanjem karaktera `z` i `k` koji zajedno čine čvor stabla (tipa `Node_S`) koji nema reprezentujući karakter, ekvivalentni broj pojavljivanja mu je 9, i sadrži reference na čvorove ispod sebe (decu).

Takav čvor (bez karaktera) se stavlja u privremeni niz, gde čeka da bude združen sa drugim takvim čvorom, ili sa novim karakterom. U oba slučaja, iskorišćeni čvor se briše; a novodobijeni čvor se dodaje u privremeni niz.
Nakon navedenog združivanja, dva čvora sa najmanjim brojem pojavljivanja su:

- Novodobijeni čvor sa 9 pojavljivanja
- Karakter `m` sa 24 pojavljivanja

Algoritam se zatim ponavlja u krug:
Združivanjem ta dva čvora dobija se novi čvor kome je ekvivalentni broj pojavljivanja jednak 33, ...
Stablo od pomenutih čvorova je na slici 3a.

Pogodno je primetiti da su dva najređa čvora uvek jedna od sledećih mogućnosti:

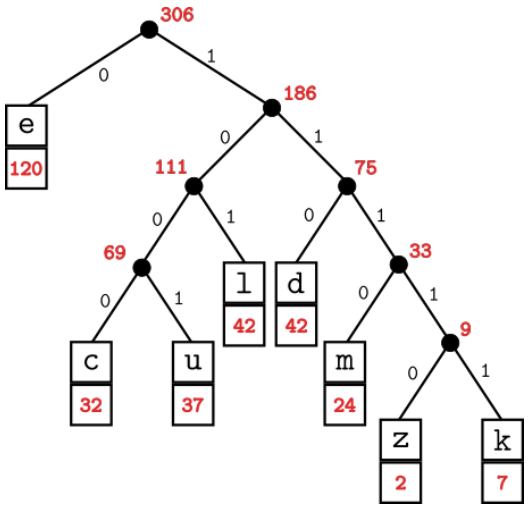
1. **Dva nova karaktera** ukoliko su brojevi pojavljivanja sledeća dva neiskorišćena karaktera manja od broja pojavljivanja svakog čvora u privremenom nizu
2. **Dva čvora iz privremenog niza** ukoliko su brojevi pojavljivanja čvorova u privremenom nizu (posmatrajući najmanja dva) manji od broja pojavljivanja najređeg neiskorišćenog karaktera
3. **Jedan novi karakter i jedan čvor iz privremenog niza** u ostalim slučajevima

U cilju određivanja koji elementi će biti združeni, dovoljno je izvršiti do dva poređenja čvorova, i poređenja dužina privremenog niza sa 2, i broja preostalih elemenata ulaznog niza sa 2 (očigledno je da je prva mogućnost moguća samo ako su preostala barem 2 ulazna elementa, i slično važi za mogućnost 2).

Imajući navedene mogućnosti u vidu, moguće je privremeni niz čuvati odvojeno od ulaznih podataka, i sortirati ga nezavisno. Time će privremeni niz imati značajno manje elemenata pa je proces sortiranja brži. Pomenuto sortiranje privremenog niza se može obavljati bilo kad, pre grananja na navedene mogućnosti, ili nakon, kada je novoformirani čvor ubačen u niz.

Dobijanje koda na osnovu stabla

Postupak za dobijanje koda simbola na osnovu stabla je izuzetno jednostavan. Kôd simbola se čuva u pomoćnom nizu. Kretanjem kroz stablo od korenog čvora (*root*) do lista u kome se nalazi simbol dobija se kôd, tako što se pri svakom skretanju levo na kôd dodaje - `0`, a pri svakom skretanju desno na kôd dodaje - `1`. Ovo je ilustrovano sledećim primerom.



(a) Primer stabla

Simbol	Kod
e	0
c	1000
u	1001
l	101
d	110
m	1110
z	11110
k	11111

(b) Rezultat dekodovanja

Slika 3: Primer dobijanja koda na osnovu stabla

Implementacija

Procedura u kojoj je ovo implementirano deklarisana je u fajlu `huffman_tree.inc`, dok se definicija nalazi u fajlu `huffman_tree.asm`. Algoritam će biti objašnjen na odabranim delovima koda iz fajla `huffman_tree.asm`.

```
PrintCodes PROC,\
FirstNode: PTR Node_S,\ ; // First Node pointer
Dist: DWORD,\ ; // Distance from first element in Nodes array
Code: PTR BYTE,\ ; // Array containing code for a leaf
Index: DWORD ; // Current index in Code array
```

Deo koda iznad predstavlja zaglavlje procedure. `FirstNode` sadrži adresu na kojoj se nalazi prvi čvor u nizu koji sadrži stablo. `Dist` predstavlja udaljenost trenutnog čvora od prvog elementa u nizu. Pri prvom pozivu procedure, ovaj parametar pokazuje na kraj niza, gde je smešten koreni čvor (*root*). `Code` sadrži adresu pomoćnog niza koji čuva kôd. `Index` predstavlja indeks u pomoćnom nizu `Code`, ukazuje na sledeću lokaciju za upis.

```
; // Initializing registers
; // esi - pointer to current node
; // edi - pointer to char in string that holds the code
mov esi, DistBytes
add esi, FirstNode
mov edi, Code
add edi, Index

; // Incrementing index
mov ecx, Index
inc ecx
```

Deo koda iznad predstavlja inicijalizaciju registara koji se koriste u proceduri. U registru `esi` nalazi se adresa trenutnog čvora, dok se u registru `edi` nalazi adresa elementa u nizu `Code` u koji se upisuje. U registru `ecx` čuva se indeks narednog elementa pri narednom pozivu procedure `PrintCodes`.

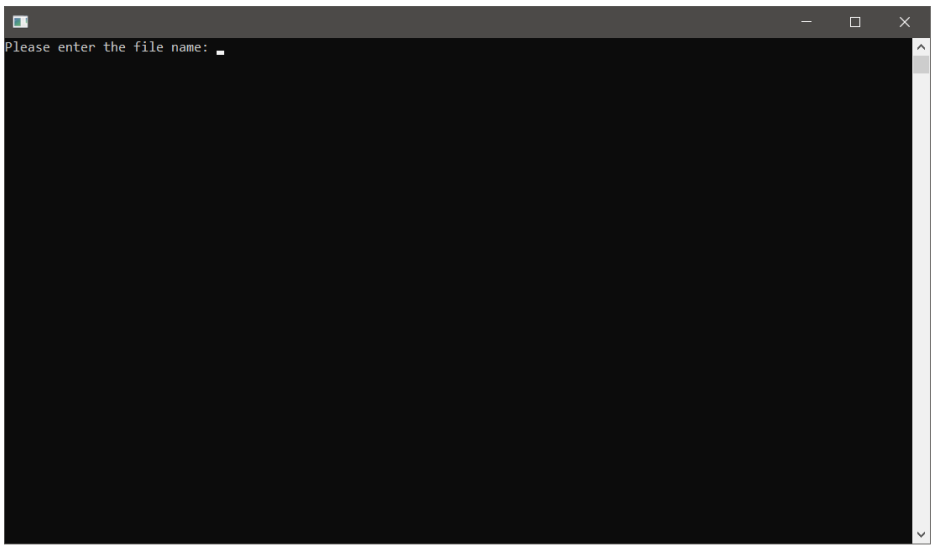
```
; // Checking if left child exists
mov al, BYTE PTR[esi + 2]
.IF(al != NUL)
mov BYTE PTR[edi], '0'
INVOKE PrintCodes, FirstNode, eax, Code, ecx
.ENDIF

; // Checking if right child exists
mov al, BYTE PTR[esi + 3]
.IF(al != NUL)
mov BYTE PTR[edi], '1'
INVOKE PrintCodes, FirstNode, eax, Code, ecx
.ENDIF
```

Deo koda iznad služi za jednostavno kretanje kroz stablo. Naime, na početku se proverava da li postoji levo dete trenutnog čvora. Ako postoji, na kôd se dodaje `0`, a zatim se ponovo poziva ista procedura za to levo dete. Ako ovo nije ispunjeno isto se radi za desno dete, s tim što se u slučaju postojanja na kôd dodaje `1` . Kao što se može videti, ovde je korišćena rekurzija, pa je izuzetno važno sačuvati registre na stek na početku procedure.

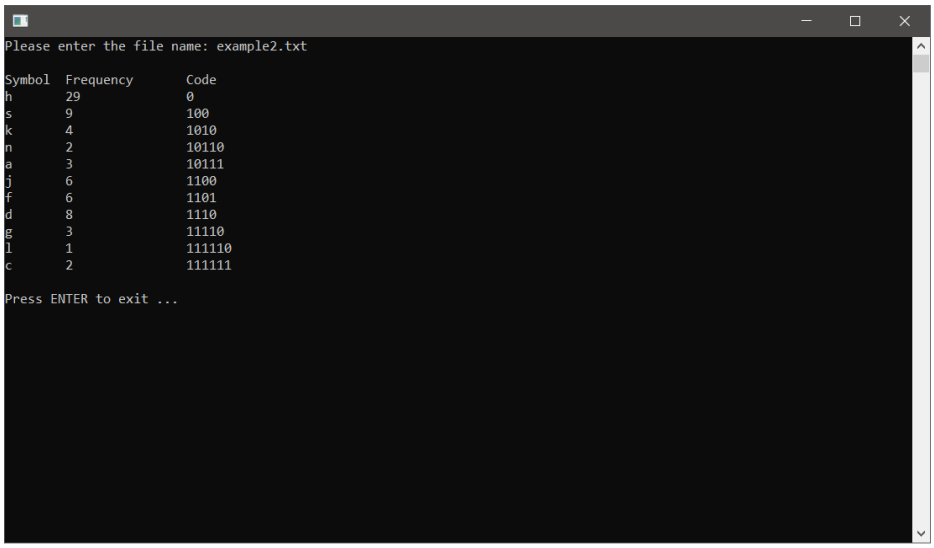
Nakon ovoga se na kôd dodaje karakter za kraj stringa - `'\0'`. Kôd koji je smešten u `Code` štampa se jedino ako trenutni čvor predstavlja list u stablu.

Uputstvo za korišćenje programa

Program se vrlo jednostavno koristi. Nakon pokretanja korisnika dočekuje prozor prikazan na slici 4. Program zahteva ime tekstualne datoteke nad kojom će biti izvršeno *Huffman*-ovo kodovanje. Nakon unosa imena datoteke, pritiskom na `ENTER` započinje izvršavanje programa. Program ispisuje rezultate na konzoli. Primer ispisa rezultata se nalazi na slici 5. Za kraj izvršavanja programa treba ponovo pritisnuti `ENTER`.



Slika 4: Konzola nakon pokretanja programa



Slika 5: Primer ispisanog rezultata nakon završenog kodovanja