پروژه ی پایانترم ساختمان داده

طراحی یک assembler

زبان مجاز : ازاد

روز تحویل : ۷ بهمن

حتما میدونید که وقتی یک کدی را داخل کامپیوتر مینویسید این کد باید اول کامپایل بشه یعنی اینکه باید به زبان سطح پایین اسمبلی تبدیل بشه.

این عملیات رو کامپایلر برای ما انجام میده که مثلا کامپایلر gcc از معروف ترین کامپایلر ها برای زبان c هست.

اما کد اسمبلی ایجاد شده هنوز هم قابلیت اجرا شدن روی cpu کامپیوتر شما را نداره به دلیل اینکه cpu فقط و فقط ۰و۱ میفهمه.

پس بعد از عملیات کامپایل کد شما باید توسط assembler به ۰و۱ تبدیل بشه که راه های زیادی برای اون وجود داره.

هدف از این پروژه طراحی یک ساختار و ارائه ی یکی از راه حل های ممکن واسه ی انجام این کاره.

برای انجام این کار از ساختمان داده ی درختی به اسم درخت هافمن استفاده میشه.

دو نکته در این الگوریتم حائز اهمیت هست:

۱ – ما باید با کمترین تعداد ممکن ۰و۱ زبان اسمبلی رو به صفر و یک تبدیل کنیم.

۲ – تبدیل ما نباید جوری باشه که باعث ابهام بشه به مثال زیر دقت کنین:

فرض کنید که چهار کاراکترd,c,b,a موجود هستند و کدهای طول متغیر متناظر با آن‌ها به ترتیب ۰۰، ۰۱، ۰ و ۱ است. این کدگذاری موجب ابهام می‌شه چون کد تخصیص یافته بهc ، پیشوند کدهای تخصیص یافته به a و b است. اگر رشته فشرده شده ۰۰۰۱ باشه، خروجی که از حالت فشرده خارج شود امکان داره cccd یا ccb یا acd یا ab باشه.

دو بخش اصلی مهم در کدگذاری هافمن وجود داره :

1. ساخت درخت هافمن از کاراکترهای ورودی
2. پیمایش درخت هافمن و تخصیص کد به کاراکترها

برای اشنایی با این الگوریتم و نحوه ی کارکرد اون به لینک زیر مراجعه کنین :

https://www.youtube.com/watch?v=co4_ahEDCho&app=desktop

چند نکته ی مهم :

۱- برای انجام این پروژه به **هیچ** دانشی از زبان اسمبلی نیاز ندارید در واقع آن را مثل یک فایل text در نظر بگیرید و با توجه به ویدیو ی اموزشی روی آن پردازش های لازم را انجام دهید .

۲ – ورودی مسئله در هنگام تحویل پروژه یک فایل text است که حاوی کد اسمبلی میباشد و برنامه ی شما باید قابلیت خواندن از این فایل و انجام بقیه کار ها را داشته باشد.

۳- کد شما باید برای **هر فایلی** جواب دهد **نه فقط** نمونه کدی که در صورت پروژه اورده شده است.

۴ – خروجی کد شما نیز باید یک فایل text باشد که **فقط** حاوی ۰/۱ است.

۵ – هر خط زبان اسمبلی را در یک خط جدا به زبان ۰و۱ تبدیل کنید به عبارتی به ازای هر خط ورودی شما باید یک خط خروجی چاپ کنید و همه ی خط ها را دنبال هم چاپ **نکنید**.

به طور مثال کد زیر یک نمونه کد ورودی است که در قالب فایل text در اختیار شما قرار میگیرد :

** دقت کنید که space ها هم خود یک کاراکتر هستند و باید انها را هم در هنگام کد نویسی لحاظ کنید **

```
main:
                addi $sp, $sp, -16
                sw $ra, 0($sp)
                sw $s0, 4($sp)
                sw $s1, 8($sp)
                sw $s2, 12($sp)
                la $a0, firstMatrix
                jal print_matrix
                jal print_new_line
                la $a0, secondMatrix
                jal print_matrix
                la $a0, firstMatrix
                la $a1, secondMatrix
                la $a2, newMatrix
                jal matrix_mult
```

```
        jal print_new_line
        jal print_new_line
        la $a0, newMatrix
        jal print_matrix
        li $v0, 10
        syscall

        lw $s2, 12($sp)
        lw $s1, 8($sp)
        lw $s0, 4($sp)
        lw $ra, 0($sp)
        addi $sp, $sp, 16
        jr $ra

matrix_mult:
        addi $sp, $sp, -32
        sw $ra, 0($sp)
        sw $s0, 4($sp)
        sw $s1, 8($sp)
        sw $s2, 12($sp)
        sw $s3, 16($sp)
        sw $s4, 20($sp)
        sw $s5, 24($sp)
        sw $s6, 28($sp)
        or $s0, $s0, $a0
        or $s1, $s1, $a1
        or $s2, $s2, $a2
        ori $s3, $s3, 0
        ori $s4, $s4, 0
        ori $s5, $s5, 0
        ori $s6, $s6, 4

matrix_mult_loop1:
        li $s4, 0

matrix_mult_loop2:
        li $s5, 0
        li.d $f6, 0

matrix_mult_loop3:
        mult $s3, $s6
        mflo $t0
        add $t1, $t0, $s5
```

```
sll $t1, $t1, 3
add $t2, $t1,$s0
l.d $f0, 0($t2)
mult $s5, $s6
mflo $t0
```