به نام خدا

توضیحات اجمالی پروژه

موضوع: مرتب سازی اعداد دهدهی با الگوریتم مرتب سازی حبابی

در این پروژه قصد داریم برنامه‌ای با استفاده از اسمبلر پردازنده (x۸۰۸۶) پیاده سازی کنیم به طوری که ابتدا آرایه‌ای شامل اعدادی بر مبنای ده را به عنوان ورودی در سگمنت داده قرار می‌دهیم سپس با استفاده از دستوراتی که در سگمنت کد تعریف می‌کنیم، الگوریتم مرتب‌سازی حبابی را روی آرایه اجرا کرده و آرایه را به ترتیب نزولی (از عضو بزرگ به کوچک) مرتب می‌کنیم.

تعریف الگوریتم مرتب سازی حبابی ، پیچیدگی و نحو اجرای آن روی یک آرایه

الگوریتم مرتب‌سازی حبابی (Bubble sort) یک الگوریتم مرتب‌سازی ساده‌است که آرایه ورودی را پشت سرهم پیمایش می‌کند تا هر بار عناصر مجاورهم را با هم مقایسه کند و اگر در جای نادرست بودند جابه‌جایشان (Swap) کند. در این الگوریتم این کار باید تا زمانی که هیچ جابه‌جایی در آرایه رخ ندهد، ادامه یابد و در آن زمان آرایه مرتب شده‌است. این مرتب‌سازی از آن رو حبابی نامیده می‌شود که هر عنصر با عنصر کناری خود مقایسه شده و درصورتی که از آن کوچک‌تر باشد با آن جا به جا می شود و این کار همچنان پیش می‌رود تا کوچک‌ترین عنصر به آخر آرایه برسد و دیگر عناصر نیز به ترتیب در جای خود قرار گیرند (یا به سمت چپ بروند یا به سمت راست آرایه رانده شوند) این عمل همانند پویش حباب به بالای مایع است.

این مرتب‌سازی از آن رو که برای کار با عناصر آن‌ها را با یکدیگر مقایسه می شوند، یک مرتب‌سازی سنجشی است.

با فرض داشتن nعضو در آرایه، در بدترین حالت n(n−۱)/۲ مقایسه لازم خواهد بود لذا بدترین زمان اجرا و پیچیدگی متوسط مرتب‌سازی حبابی از مرتبه O(n۲) می‌باشند

سورس کد برنامه نوشته شده به زبان اسمبلی (x۸۰۸۶)

```
org 100h
.data

array  db 1,4,2,3
length  dw 4

.code

    mov cx,length
    dec cx
iteronarray:
    mov bx,cx
    mov si,0

cmponarray:
    mov al,array[si]
    mov dl,array[si+1]
    cmp al,dl

    jnc noswap
    mov array[si],dl
    mov array[si+1],al


noswap:
    inc si

    dec bx
    jnz cmponarray

    loop iteronarray
```

توضیح خط به خط پیاده سازی به زبان اسمبلی (x۸۰۸۶)

۱. آدرس دهی آفست برنامه

۲. پیاده سازی سگمنت داده

۴. تعریف و مقدار دهی آرایه ای از اعداد دهدهی به صورت مجموعی از بایت ها (هر عضو یک بایت)

۵. تعریف کاردینال (اندازه) آرایه با یک کلمه (دو بایت)

۷. پیاده سازی سگمنت کد

۹. مقدار کاردینال آرایه را در ثبات شمارشگر(CX) بار می کنیم

۱۰. یک واحد از ثبات شمارشگر(CX) را کم می کنیم تا تعداد مقاسیه لازم در هر پیمایش را به دست آوریم (برای مثال برای آرایه ۴ عضوی بالا ۳ مقایسه لازم است)

۱۱. پیاده سازی دستور پیمایش روی اعضای آرایه

۱۲. بار کردن تعداد مقایسه لازم در هر پیمایش در ثبات BX

۱۳. بار کردن مقدار ایندکس عضو اول آرایه یعنی صفر در ثبات source index

۱۵. پیاده سازی دستور مقایسه دو عنصر مجاورهم

۱۶. بار کردن مقدار عنصر اول آرایه در ثبات AL

۱۷. بار کردن مقدار عنصر بعدی در ثبات DL

۱۸. مقایسه مقادیر دو عنصر (در صورتی که عنصر دومی بزرگتر از عنصر اولی باشد مقدار carry flag برابر یک خواهد شد)

۲۰. در صورتی که مقدار بارگذاری‌شده در ثبات AL (عنصر اولی) بزرگ‌تر از مقدار بارگذاری‌شده در ثبات DL (عنصر دومی) باشد، به دلیل استفاده از دستور مقایسه(CMP) در بالا، رقم نقلی تولید شده و مقدار پرچم نقلی (CF) به ۱ تغییر می‌کند در نتیجه با اجرای دستورJNC پرش به دستور noswap اتفاق می افتد

در واقع رخ دادن پرش به این دلیل است که در صورتی که عنصر اولی از دومی بزرگ تر باشد یا باهم برابر باشند از عمل جابه جایی(swap) جلوگیری کند زیرا در ابتدا فرض کرده ایم که مرتب سازی ما نزولی باشد

۲۱. اما در صورتی که عنصر اولی کوچکتر از دومی بود باید این دو عنصر را باهم جابه جا(swap) کنیم

۲۵. پیاده سازی دستور noswap (در صورتی جابه جایی رخ ندهد از دستور مقایسه مستقیماً به این دستور پرش خواهد کرد تا ادامه پیمایش روی سایر اعضا نیز دنبال شود)

۲۶. مقدار ثبات source index یک واحد افزایش می دهیم تا ایندکس عنصر بعدی را برای مقایسه به دست آوریم

۲۸. با انجام مقایسه بالا یک واحد از تعداد مقایسه های لازم را کم میکنیم

۲۹. تا زمانی که مقایسه لازم باشد یعنی مقدار ثبات(BX) مخالف صفر باشد می بایست پیمایش و مقایسه عناصر ادامه داشته باشد یعنی پرش به دستور مقایسه رخ دهد مگر

توضیح خط به خط پیاده سازی به زبان اسمبلی (x۸۰۸۶)

زمانی که مقایسه ای لازم نباشد یعنی تعداد مقایسه لازم برابر صفر یا به عبارتی دیگر مقدار ثبات(BX) برابر صفر شود (روند فوق با استفاده از دستور JNZ پیاده سازی کرده ایم)

زمانی که پیمایش روی آرایه به عنصر آخر رسید و دستور مقایسه اجرا شد میدانیم حال عنصر آخر در مکان صحیح قرار دارد یا به عبارتی کوچکترین عضو آرایه است پس حلقه را برای سایر اعضا ادامه میدهیم تا به عنصر اول برسیم و این را به وسیله دستور حلقه(loop) پیاده سازی میکنیم

۳۱. با استفاده از دستور حلقه(loop) ، پیمایش مجدداً روی آرایه با کاهش یک واحد از مقدار ثبات شمارشگر (CX) ادامه می‌یابد. این فرآیند تا زمانی ادامه خواهد داشت که مقدار CX به صفر برسد، به این معنا که آرایه به‌طور کامل مرتب شده است

برای مثال بالا، جدول تغییر مقادیر ثبات‌های مورد استفاده در برنامه را تشکیل داده‌ایم:

AX	0000	0001	0004	0002	0004	NULL	NULL	NULL	NULL	NULL
BX	0000	0003	0002	0001	0000	0002	0001	0000	0001	0000
CX	0045	0004	0003	0002	0001	0000	0004	NULL	NULL	NULL
DX	0000	0004	0002	0003	0002	0003	NULL	NULL	NULL	NULL
SI	0000	0001	0002	0003	0000	0001	0002	0000	0001	0000
CF	0	1	0	1	0	0	NULL	NULL	NULL	NULL
ZF	0	1	0	1	0	1	0	NULL	NULL	NULL