طراحی سیستم های میکروپروسسوری
فاز دو Going Faser
علی یداللهی
شماره دانشجویی : ۴۰۰۱۰۲۲۳۳

۱

در این فاز از پروژه هدف ما به حداقل رساندن دسترسی ها به حافظه اصلی و استفاده بهتر از Cache بود.
برای این منظور از دو روش استفاده شد.

- transpose کردن ماتریس دوم و ضرب ماتریس اول در آن: برای این کار حافظه جدیدی را به ماتریس ترانهاده اختصاص دادم سپس توسط تابعی با اسم transpose ماتریس اولیه را ترانهاده کرده و در ماتریس جدید ریختم.سپس ماتریس A را در ماتریس Btranspose ضرب می کنیم.فرآیند ضرب تا حد زیادی مشابه تابع matrix mult ptr reg است و تنها تفاوت این است که به جای ضرب هر سطر ماتریس اول در هر ستون ماتریس دوم هر سطر ماتریس اول در سطر ماتریس دوم ضرب می شود.
- ضرب بلوکی ماتریس ها: برای پیاده سازی ضرب بلوکی از الگوریتم آورده شده در دستور کار استفاده شده است. به این صورت که ابتدا بلوک ها را همانند درایه های ساده فرض کرده و بلوک هایی که باید در هم ضرب شوند را مشخص می کنیم. سپس با استفاده از همان الگوریتم ضرب ماتریسی در قسمت های قبل دو بلوک را در هم ضرب می کنیم.

در ادامه به ازای سه مقدار $n = 1024, n = 2048, n = 4096$ نتایج دو تابع جدید با سریع ترین حالت سری قبل (استفاده از پوینتر و رجیستر) مقایسه شده اند.

- n = 1024

  block size = 8

function	1	2	3	4	5
mult-ptr-reg	8.54	8.44	8.40	8.69	8.51
mult-transpose	3.03	3.03	3.00	3.08	3.07
mult-block	2.64	2.62	2.72	2.72	2.63

- n = 2048

  block size = 8

function	1	2	3	4	5
mult-ptr-reg	130.58	131.54	128.21	130.51	130.21
mult-transpose	24.49	24.74	24.52	24.55	24.62
mult-block	22.00	21.75	21..73	21.41	21.87

-

n = 4096

block size = 8

function	1	2	3	4
mult-ptr-reg	1372.18	1380.57	1383.39	1390.03
mult-transpose	202.26	202.05	202.43	202.77
mult-block	180.28	194.32	195.97	183.82

در ادامه به ازای block size های مختلف نتایج آورده شده اند:

- n = 1024

block size	1	2	3	4	5
16	2.36	2.35	2.33	2.28	2.57
32	2.44	2.45	2.71	2.65	2.44
64	2.89	2.86	2.91	2.91	2.89
128	2.91	2.91	2.94	2.94	2.91
256	2.96	2.91	2.93	2.95	2.88

- n = 2048

block size	1	2	3	4	5
16	22.25	22.36	20.22	20.23	20.22
32	25.55	23.73	24.44	23.97	24.19
64	22.86	23.53	22.51	23.66	23.15
128	23.18	23.69	23.81	23.02	22.99
256	23.04	23.33	29.29	22.80	23.38

- n = 4096

block size	1	2
16	219.89	222.77
32	242.37	237.01
64	219.29	222.19
128	224.15	216.76
256	244.81	243.71

می توان دید که زمان اجرا کاهش زیادی دارد. در مقایسه روش ترانهاده و بلوکی هم می توان دید که در روش بلوکی زمان اجرا اندکی کمتر از روش ترانهاده است. البته وقتی $n = 4096$ است فقط به ازای $\text{block size} = 8$ روش بلوکی بهتر از ترانهاده است و با افزایش block size زمان اجرا کمی بیشتر از روش ترانهاده می شود.
در ادامه زمان میانگین برای هرکدام از حالات آورده شده است. در میانگین گیری برخی داده های پرت حذف شده اند.

$\text{block size} = 8$

function	1024	2048	4096
mult-ptr-reg	8.47	130.71	1381.54
mult-transpose	3.04	24.58	202.38
mult-block	2.67	21.75	188.59

میانگین زمان اجرا به ازای block size های مختلف:

block size	1024	2048	4096
16	2.33	21.06	221.33
32	2.54	24.37	239.69
64	2.89	23.14	221.03
128	2.92	23.34	220.45
256	2.94	23.14	244.26