به نام خدا

برنامه‌نویسی چندهسته‌ای

تمرین سوم

مهلت تا ۱۳۹۸/۲/۱۶

نمره تمرین: 100 واحد

قسمت اول

مطابق اسلاید ۸ در openmp-part3 هیستوگرام را با روش خواسته‌شده پیاده‌سازی کنید.(استفاده از هیستوگرام‌های محلی و ادغام نهایی آنها)

ورودی بین اعداد ۰ تا ۵۰۰۰ بدهید و تعداد عددها $۱۰^۷$ باشد

قسمت دوم

در این قسمت بیشتر با بخش Microarchitecture Exploration در Intel Parallel Studio برای مشاهده اثر کد نامناسب در شاخص Bad speculation آشنا می‌شویم.

همان‌طور که در درس معماری آشنا شدید، پردازنده‌ها واحدی به اسم Branch Predictor دارند. این واحد تلاش می‌کند که حدس بزند که یک پرش چه راهی می‌رود قبل از اینکه به طور قطعی شناخته‌شود. هدف بهبود جریان دستورات در Pipeline است. حال اگر کدی داشته باشیم که دارای شاخه‌های تو در تو باشد و به صورت تصادفی انجام شوند پیشرفته‌ترین مکانیزم‌های branch predection نیز توانایی پیش‌بینی ندارند و به اشتباه پیش‌بینی می‌کنند. پیش‌بینی غلط به معنی Flush پایپلاین است که باعث کاهش کارایی می‌شود.

شاخص Bad speculation این مورد را مورد بررسی قرار می‌دهد. در دنیا واقعی همیشه کتابخانه‌های مورد استفاده ممکن است به خوبی پیاده‌سازی نشده باشند و ما توانایی تغییر در آن‌ها را نداشته باشیم. برای همین جهت بهبود عملکرد باید تغییری در ورودی‌ها ایجاد کنیم.

Header File داده شده به اسم worst_categorizer.h دارای تابعی به نام categorize است که اعداد بین ۰ تا ۴۰۰۰ گرفته و به ۸۵ دسته تقسیم می‌کند

Prototype این تابع به شکل زیر است:

```
int categorize(int a)
```

برای انجام بخش‌های زیر از تابع بالا استفاده کنید.

بخش اول

ابتدا ارائه‌ای به طول $۷*۱۰^۸۶$ را با اعداد تصادفی بین ۰ تا ۴۰۰۰ پرکنید

با استفاده از تابع categorize اعداد را دسته‌بندی کنید(به‌صورت سریال)

با استفاده از Intel Vtune Amplifier میزان miss prediction را گزارش دهید

بخش دوم

آرایه را با اعداد متوالی بین ۰ تا ۴۰۰۰ پرکنید و با قسمت قبل مقایسه کنید

بخش سوم

ابتدا ارائه‌ای به طول $۷*۱۰^۸۷$ را با اعداد تصادفی بین ۰ تا ۴۰۰۰ پرکنید

با استفاده از تابع categorize اعداد را دسته بندی کنید(به‌صورت موازی)

زمان اجرا را با بخش اول مقایسه کنید

با استفاده از Intel Vtune Amplifier میزان miss prediction را گزارش دهید

بخش چهارم

حال آرایه را با اعداد متوالی بین ۰ تا ۴۰۰۰ پرکنید و با قسمت قبل مقایسه کنید

بخش پنجم

راهی بیابید که بتوان بدون اینکه تصادفی بودن ارایه را از بین ببرید، بتوانید تسریع مشاهده کنید . در بخش اول بجز قسمت دسته‌بندی و پر کردن آرایه می‌توانید قسمت اضافه شده را به صورت موازی انجام دهید.

آیا راه پیشنهادی در قسمت بخش سوم نیز تسریع می‌گیرد؟

راهنمایی: استفاده از مرتب‌سازی بهینه می‌تواند به کاهش miss prediciton کمک کند

توجه: در تمامی قسمت‌ها مجاز به تغییر header file نیستید

نحوه تحویل

کد هر بخش را به‌صورت تابعی با نام همان بخش قرار دهید

زمان اجرای هر قسمت

تصاویر microarchitecture exploration مربوط به هر بخش مانند شکل زیر:

Elapsed Time: 15.439s

Clockticks:	52,707,600,000	
Instructions Retired:	73,695,600,000	
CPI Rate:	0.715	
MUX Reliability:	0.786	
Retiring:	23.7%	of Pipeline Slots
Front-End Bound:	100.0%	of Pipeline Slots
Front-End Latency:	36.6%	of Pipeline Slots
ICache Misses:	0.0%	of Clockticks
ITLB Overhead:	0.0%	of Clockticks
Branch Resteers:	3.5%	of Clockticks
DSB Switches:	4.3%	of Clockticks
Length Changing Prefixes:	0.0%	of Clockticks
MS Switches:	0.0%	of Clockticks
Front-End Bandwidth:	71.6%	of Pipeline Slots
Front-End Bandwidth MITE:	7.5%	of Clockticks
Front-End Bandwidth DSB:	70.4%	of Clockticks
Front-End Bandwidth LSD:	0.0%	of Clockticks
(Info) DSB Coverage:	90.1%	
(Info) LSD Coverage:	0.0%	
Bad Speculation:	6.5%	of Pipeline Slots
Branch Mispredict:	6.5%	of Pipeline Slots
Machine Clears:	0.0%	of Pipeline Slots
Back-End Bound:	0.0%	of Pipeline Slots
Memory Bound:	0.0%	of Pipeline Slots
L1 Bound:	2.6%	of Clockticks
L2 Bound:	N/A with HT on