

## برنامهنويسي چندهستهاي

تمرین سوم

مهلت تا ۱۳۹۸/۲/۱۶

نمره تمرين: 100 واحد



### قسمت اول

مطابق اسلاید ۸ در openmp-part3 هیستوگرام را با روش خواسته شده پیاده سازی کنید. (استفاده از هیستوگرام های محلی و ادغام نهایی آنها)

ورودی بین اعداد  $\cdot$  تا ۵۰۰۰ بدهید و تعداد عددها  $^{\vee}$  باشد

#### قسمت دوم

در این قسمت بیشتر با بخش Microarchitecture Exploration در Intel Parallel Studio برای مشاهده اثر کد نامناسب در شاخص Microarchitecture Exploration برای مشاهده اثر کد نامناسب در شاخص آشنا می شویم.

همان طور 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 شناخته شود. 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 پیشرفته ترین مکانیزمهای branch predection نیز توانایی پیش بینی ندارند و به اشتباه پیش بینی می کنند. پیش بینی غلط به معنی Flush پایپلاین است که باعث کاهش کارایی می شود.

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 می کند

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 ۲۰۰۰\*۷ را با اعداد تصادفی بین ۰ تا ۴۰۰۰ پرکنید

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

با استفاده از Intel Vtune Amplifier ميزان miss prediction را گزارش دهيد

## بخش دوم

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

## بخش سوم

ابتدا ارائهای به طول ۱۰۸۷\*۷ را با اعداد تصادفی بین ۰ تا ۴۰۰۰ پرکنید

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

زمان اجرا را با بخش اول مقایسه کنید

با استفاده از Intel Vtune Amplifier ميزان miss prediction را گزارش دهيد

#### بخش چهارم

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

#### بخش ينجم

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

آیا راه پیشنهادی در قسمت بخش سوم نیز تسریع می گیرد؟

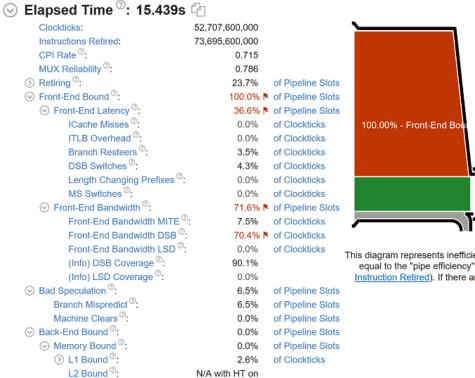
راهنمایی: استفاده از مرتبسازی بهینه می تواند به کاهش miss prediciton کمک کند

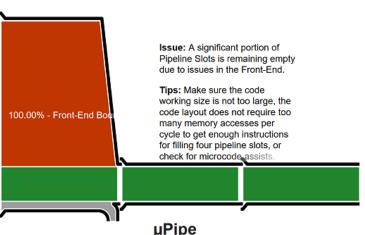
# توجه: در تمامی قسمتها مجاز به تغییر header file نیستید نحوه تحویل

کد هر بخش را به صورت تابعی با نام همان بخش قرار دهید

زمان اجرای هر قسمت

تصاویر microarchitecture exploration مربوط به هر بخش مانند شکل زیر:





This diagram represents inefficiencies in CPU usage. Treat it as a pipe with an output flow equal to the "pipe efficiency" ratio: (Actual Instructions Retired)/(Maximum Possible Instruction Retired). If there are pipeline stalls decreasing the pipe efficiency, the pipe shape gets more narrow.