دانشگاه صنعتی شریف

دانشکده مهندسی برق

درس طراحی سیستمهای میکرو پروسسوری (۲۵۷۷۱)

مجموعه آزمایش‌های بهره گیری از قابلیت‌های پردازنده‌ها

مرحله‌ی اول : پوینتر و رجیستر در زبان C

تهیه کنندگان:

محمّدرضا موحّدین

علیرضا عباسیان

به نـــــام خـــــــدا

مقدمه

هدف از این مجموعه آزمایش‌ها که در چند فاز ارائه می‌شود، فعال‌سازی و بهره‌گیری از قابلیت‌های گوناگون پردازنده‌ها است که از مجموعه بخش‌های Going Faster کتاب Computer Organization & Design: The HW/SW Interface, 6th Edition الهام گرفته شده اند. این بخش‌های کتاب فوق در یک فایل جداگانه در اختیار شما قرار می‌گیرد.

در این آزمایش‌ها، عملیات ضرب دو ماتریس مربعی در نظر گرفته شده و در مراحل گوناگون تلاش می‌شود زمان اجرای این ضرب بهبود یابد. این مراحل شامل استفاده از پوینتر و رجیستر در زبان C، بکارگیری حافظه‌ی Cache، بکارگیری قابلیت‌های اجرای چندگانه‌ی پردازنده با Loop Unrolling، استفاده از دستورات برداری از قبیل SSE و AVX و نهایتاً استفاده از چند هسته‌ی پردازنده می‌باشند.

مرحله‌ی صفر: کد مرجع ضرب

کد مقابل به عنوان کد مرجع و نقطه‌ی شروع آزمایش‌ها مورد استفاده قرار می‌گیرد که فرایند ضرب ماتریس را در سه حلقه‌ی ساده و سر راست اجرا می‌کند. زمان اجرای این کد برای سایز مشخصی از ماتریس (متغیر n) به عنوان مبدأ مقایسه‌ی بهبود کیفیت کدهای آتی است. همچنین این کد برای صحت سنجی کدهای آینده مورد استفاده قرار خواهد گرفت. البته این کار باید با مقدار کوچک n (مثلاً حدود ۲۰ تا ۱۰۰) صورت گیرد تا زمان صحت سنجی بیش از اندازه نشود.

```
void matrix_mult_0
(int n, double* a, double* b, double* c){
  int i, j, k;
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++){
      c[i*n+j] = 0.0;
      for(k = 0; k < n; k++)
        c[i*n+j] += a[i*n+k] * b[k*n+j];
    }
}
```

مرحله‌ی اول: استفاده از پوینتر و رجیستر در زبان C

در اولین قدم جهت بهبود سرعت اجرای ضرب ماتریس‌ها، کد مقابل را در نظر بگیرید. در این تابع، دیگر از آدرس‌دهی آرایه‌ها توسط اندیس استفاده نشده بلکه پوینترها به جای آنها به کار رفته‌اند. همچنین تلاش شده است که کامپایلر متغیرهای پر استفاده را به رجیسترهای داخلی پردازنده اختصاص دهد.

```
void matrix_mult_1
(int n, double* a, double* b, double* c){
  register double cij;
  register double *at, *bt;
  register int i, j, k;
    for (i = 0; i < n; i++, a+ = n)
      for (j = 0; j < n; j++, c++) {
        cij = 0;
        for(k = 0, at = a, bt = &b[j];
            k < n; k++, at++, bt += n)
              cij += *at * *bt;
        *c = cij;
      }
}
```

۱-۱- دو تابع فوق را با مقادیر مختلف n اجرا و زمان اجرا را مقایسه کنید. قطعه کد صفحه‌ی بعد می‌تواند برای این منظور مورد استفاده قرار گیرد.

۱-۲- اثر استفاده از کلمه کلیدی register در تابع دوم را با حذف آن بررسی کنید.

۱-۳- تابع دوم را به ازای اندازه‌های گوناگون ماتریس یا همان متغیر n اجرا کنید. در غالب پردازنده‌ها، حوالی n = 1024 زمان اجرا یک افزایش چشمگیر دارد. علت چیست؟

نکات مهم در بررسی زمان اجرا

۱- کلیه‌ی بهینه‌سازی‌های کامپایلر (Compiler Optimizations) را قطع کنید تا فقط نتیجه‌ی اقدامات شما در نتیجه‌ی خروجی ظاهر شود.

۲- کلیه‌ی برنامه‌های جانبی از قبیل آنتی ویروس، به روز رسانی‌ها، Search Indexing و دیگر برنامه‌ها را هنگام استخراج زمان اجرای توابع متوقف کنید تا زمان اجرای تابع هر چه بیشتر دقیق باشد.

۳- زمان را در چندین اجرا استخراج و میانگین آنها را به دست آورید. زمان‌های طولانی غیر متعارف را اولاً از میانگین حذف و ثانیاً علت‌یابی نمایید.