- سوالات خود را می‌توانید از طریق تلگرام از علی جهان (@alijahan890) و فاطمه درج (@F_Dorj) بپرسید.
- یک گزارش کامل (فایل PDF) شامل شرح روش‌ها، روند حل مسئه، اسکرین شات از پروژه و نتایج نهایی را آپلود کنید.
- فایل‌های پروژه شامل کد اسمبلی(فایل s.) را همراه با گزارش پروژه در قالب یک فایل Zip به فرمت P2-studentNumber.zip بارگذاری کنید.
- پروژه‌ها تحویل آنلاین خواهند داشت پس حتما دقت داشته باشید که باید به طور کامل به کد خود مسلط باشید. تسلط شما بر پروژه، به صورت یک ضریبی بین ۰ تا ۱ بر روی نمره پروژه شما خواهد بود.

مقدمه و توضیح کلی

در سیستم‌های کامپیوتری مدرن استفاده از حافظه‌های پرسرعت مانند حافظه نهان (Cache) به عنوان واسطه‌ای میان پردازنده و حافظه اصلی (Main Memory) نقشی حیاتی در افزایش کارایی سیستم ایفا می‌کند. حافظه نهان با ذخیره‌ی موقت داده‌هایی که احتمال استفاده‌ی مجدد از آن‌ها بالاست، باعث کاهش زمان دسترسی به داده‌ها و افزایش سرعت کلی پردازش می‌شود. در درس معماری کامپیوتر با نحوه‌ی عملکرد سخت‌افزاری حافظه نهان آشنا می‌شوید؛ اما در این پروژه از درس زبان اسمبلی، هدف این است که درک شما از مفاهیم حافظه نهان با پیاده‌سازی یک شبیه‌ساز نرم‌افزاری عمیق‌تر شود.

شرح مسئله

در این پروژه، شما باید یک شبیه‌ساز حافظه نهان به زبان اسمبلی پیاده‌سازی کنید که عملکرد واقعی یک سیستم ساده حافظه نهان را شبیه‌سازی کرده و رفتار آن را در مواجهه با یک دنباله‌ی ورودی از آدرس‌های حافظه بررسی نماید. مشخصات کلی سیستم:

- Main Memory: حافظه ای ۲۵۶ بایتی با آدرس های ۰ تا ۲۵۵
- Cache: حافظه ای با تعداد مشخصی بلوک (قابل تنظیم)
- اندازه بلوک حافظه نهان: ۱ بایت (برای سادگی)
- Mapping: مستقیم (Direct Mapping) یا (Set Associative)
- Replacement Policy: الگوریتم های مختلف حافظه نهان برای جایگزینی داده ها

برای هر آدرس از دنباله ورودی برنامه باید بررسی کنید که آیا آدرس در حافظه نهان موجود است (Hit) یا نه (Miss). سپس بر اساس الگوریتم انتخابی حافظه نهان را به‌روزرسانی کند.

ورودی‌های مورد انتظار

- یک دنباله از آدرس‌های حافظه (مثلاً [۴، ۵، ۶، ۴، ۷، ۵، ۸])
- سایز حافظه نهان (تعداد بلوک ها مثلا ۸ یا ۱۶)
- انتخاب الگوریتم مدیریت حافظه نهان برای هر اجرا

نکته مهم: به ازای هر دنباله ورودی، باید بتوانید شبیه سازی را با چند الگوریتم های مختلف باید ذخیره شده و با یکدیگر مقایسه شوند

خروجی های مورد انتظار

برای هر اجرای برنامه (با الگوریتم مشخص) نمایش وضعیت هر دسترسی می بایستی مشخص باشد مانند شکل ۱:

```
Address 12 --> MISS
Address 45 --> HIT
```

شکل ۱

در انتها نیز مجموع تعداد دسترسی‌ها، تعداد Hit و Missها، درصد Hit_Rate و نام الگوریتم مورد استفاده مشخص باشد. برای مثال خروجی مشابه تصویر ۲ مطلوب خواهد بود:

```
-- Summary Report --
Input Sequence: [4,5,6,4,7,5,8]

Algorithm: FIFO
Hits: 2, Misses: 5, Hit Rate: 28.5%

Algorithm: LRU
Hits: 3, Misses: 4, Hit Rate: 42.8%

Algorithm: LFU
Hits: 4, Misses: 3, Hit Rate: 57.1%

Algorithm: MFU
Hits: 2, Misses: 5, Hit Rate: 28.5%
```

شکل ۲

الگوریتم‌های مورد نیاز

شما موظف هستید همه‌ی الگوریتم‌های زیر را پیاده‌سازی کنید:

الگوریتم	توضیح
FIFO (First In First Out)	قدیمی‌ترین ورودی از حافظه نهان خارج می‌شود.
LRU (Least Recently Used)	کم‌استفاده‌ترین داده در بازه اخیر خارج می‌شود.
MRU (Most Recently Used)	آخرین داده استفاده‌شده خارج می‌شود.
LFU (Least Frequently Used)	داده‌ای که کمترین تعداد استفاده را دارد، حذف می‌شود.
MFU (Most Frequently Used)	داده‌ای که بیشترین تعداد استفاده را دارد، حذف می‌شود.
Random Replacement	بصورت تصادفی یک بلوک حافظه نهان جایگزین می‌شود.

جدول ۱: مقایسه الگوریتم‌های مدیریت حافظه نهان

بخش امتیازی

در این مرحله می توانید یک گام به پیاده سازی در دنیای واقعی نزدیک تر شوید و از یک حافظه نهان دو سطحی L1 و L2 استفاده کنید:

- ابتدا بررسی می‌شود آیا آدرس در حافظه نهان L1 وجود دارد یا نه.
- اگر در L1 نبود، بررسی L2 انجام می‌شود.
- اگر در L2 هم نبود، داده از حافظه اصلی آورده می‌شود و در L1 قرار می‌گیرد.
- باید بتوان از الگوریتم‌های متفاوتی برای L1 و L2 نیز استفاده کرد، مثلاً LRU در L1 و FIFO در L2.

در خروجی نهایی، آمار باید برای هر سطح به‌صورت جداگانه و همچنین به‌صورت کلی گزارش شود:

- L1 Hits: 5
- L2 Hits: 2
- Misses: 3
- Global Hit Rate: 70%