پاسخ نامه تمرین ۲ معماری کامپیوتر

توجه: راه حل سوال اول با دقت گرفتن اصلاحیه می باشد

(۱)

$$T_{access} = t_1 + (1-h_1)t_2 + (1-h_1)(1-h_2)t_m$$

$$h_1(b_1) = \frac{1}{1000} e^{0.2b_1} \qquad h_2(b_2) = \frac{1}{1000} e^{0.4b_2} \qquad b_1 + b_2 = 15$$

$$T_{access} = 40 + \left(1 - \frac{1}{1000} e^{0.2b_1}\right) \times 20 + \left(1 - \frac{1}{1000} e^{0.2b_1}\right)\left(1 - \frac{1}{1000} e^{0.4(15-b_1)}\right) \times 500$$

$$Cost = 30\,b_2 + 12b_1 = 30(15-b_1) + 12b_1 = 450 - 18b_1$$

$$np.argmin\left(T_{access}(b_1) \times Cost(b_1)\right) = 0$$

$$0 \le b_1 \le 15$$

```
import numpy as np
b1=np.arange(16) #b1 = [0,1,2,....,15]
target_function=40+(1-0.001*np.exp(0.2*b1))*20+(1-0.001*np.exp(0.2*b1))*(1-0.001*np.exp(0.4*(15-b1)))*500
print(np.argmin(target_function))

0
```

بنابراین: $b_1 = 0$ و $b_2 = 15$

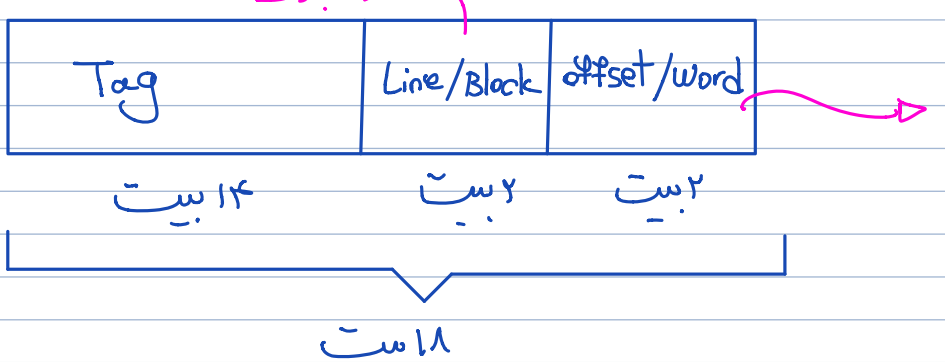
(۲) حجم حافظه اصلی ۲۵۶ کیلوکلمه ای باشد پس فضای آدرس دهی آن به تعداد بیت زیر نیاز دارد:

$$256K = 2^8 \times 2^{10} = 2^{18}$$ ← ۱۸ بیت

(۸ کیلو)

کلا ۴ بلوک داریم پس برای نشان دادن بلوک ها به $\log_2^4 = 2$ بیت نیاز داریم.

هر بلوک نیز ۴ کلمه دارد پس ۲ بیت نیاز داریم تا آدرس کلمه را مشخص کنیم.



Index به معنای حجم کل حافظه نهان می باشد که برابر با Line+offset می باشد که برابر با ۴ بیت است.

★ توضیحات تکمیلی: در این گونه سوالات معمولاً باید ابتدا تعداد بیت های کل حافظه را پیدا کنیم سپس با توجه به اطلاعات سوال باید بقیه فیلدها را مشخص کنیم. مثلا offset/word برابر با تعداد بیت های مورد نیاز برای نمایش کلمه هاست که یعنی: تعداد کلمه ها = $\log_2$ و فیلد Line/Block برابر با تعداد بیت های مورد نیاز برای نمایش تعداد بلوک ها یعنی: تعداد بلوک = $\log_2$ و فیلد Tag برابر با: کل - Line_offset باشد.

Tag = ۱۴ bits Index = ۴ bits Block = ۲ bits word = ۲ bits

(۱)

آدرس ۱۷۰ وقتی ارسال می شود، آدرس های ۱۶۸، ۱۶۹، ۱۷۰ و ۱۷۱ نیز به بلوک B۲ فرستاده می شوند.

B۲ بلوک:

w۰	۱۶۸
w۱	۱۶۹
w۲	۱۷۰
w۳	۱۷۱

اگر hit رخ داد که کاری نمی کنیم ولی اگر miss رخ داد، اگر بلوک مورد نظر خالی بود که ۴ آدرس سوای را جای گذاری می کنیم ولی اگر قبلاً اشغال شده باشد، هر ۴ آدرس قبلی پاک می شود و ۴ آدرس جدید جای آنها می شنید.

به همین ترتیب ادامه می دهیم:

۱۷۰	۲۵۷	۱۶۸	۲۴۶	۱۷۶	۱۷۵	۱۷۶	۱۷۷
✗	✗	✓	✗	✗	✗	✓	✓
۱۷۵	۱۷۶	۱۷۷	۱۷۵	۱۷۶	۱۷۷	۱۷۶	۱۷۵
✓	✓	✓	✓	✓	✓	✓	✓
۱۷۴	۱۷۳	۱۷۲	۱۷۱	۱۷۰	۱۶۹	۱۶۸	۱۶۷
✓	✓	✓	✓	✓	✓	✓	✗
۱۶۸	۱۶۵	۱۶۴					
✓	✓	✓					

Hit Rate = $\frac{۲۱}{۲۷}$ = ۷۷٪

(۲)

AA : A 10 10 ⇝ B_2W_2 : A8, A9, AA, AB miss

101: 10 0001 ⇝ B_0W_1 : 100, 101, 102, 103 miss

A8: A 1000 ⇝ B_2W_0 hit

103 : 10 0011 ⇝ B_0W_3 hit

B0 : B 0000 ⇝ B_0W_0 : B_0, B_1, B_2, B_3 miss

Hit Rate: $\frac{2}{5} = 40\%$

(۳) با توجه به شکل مشاهده می‌شود که نرخ برخورد برنامه a بیشتر از b است که نشان دهنده این است که میزان دسترسی به داده‌ها در a محلی‌تر از b است. برنامه c هم دسترسی به داده‌ها تقریباً تصادفی است.

(۴)

$$T_{access} = t_1 + (1-h_1)t_2 + (1-h_1)(1-h_2)t_3 + \underbrace{(1-h_1)(1-h_2)(1-h_3)t_4}_{\text{ضرایب } t_4} + \left[(1-h_1)(1-h_2)(1-h_3)\cdots\right]t_m$$

ب)

$$\text{ضرایب } t_j = (1-h_1)(1-h_2)(1-h_3)\cdots(1-h_{j-1}) = \prod_{i=1}^{j-1}(1-h_i) \qquad j>1$$

ج) از آنجا که به ازای هر i ام، $0<h_i<1$ بنابراین $0<1-h_i<1$، پس ضریب t_1 از ضریب هر t_j که $j>1$ است، پس کامش t_1 بیشترین سهم در کامش T_{access} دارد

د) از آنجا که در صورت ثابت ماندن h_1 بیش از هر t_j ها، عبارت $(1-h_1)$ بیش از هر $(1-h_j)$ دیگری در ضرب t_j های $j>1$ ضرب شده است، پس افزایش h_1 یا معادل کامش $(1-h_1)$ بیشترین سهم را در کامش T_{access} دارد.