

۱

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

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

$$T_{access} = \varepsilon_0 + \left(1 - \frac{1}{1000} e^{0.2b_1}\right) \times 20 + \left(1 - \frac{1}{1000} e^{0.2b_1}\right) \left(1 - \frac{1}{1000} e^{0.4(15-b_1)}\right) \times 500$$

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

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

$$0 \leq b_1 \leq 15$$

```
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))
```

بنابراین: $b_1 = 0, b_2 = 15$

۲

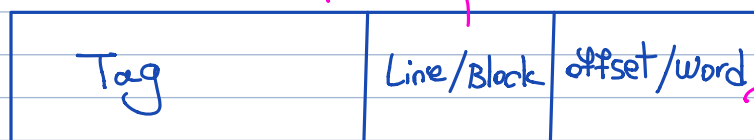
عمیق حافظه اصلی ۲۵۶ کیلوگرمی با ۲۵۶ فضای آدرس دهی آن به تعداد بیت زیر نیاز دارد:

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

که کیلو

کلمه ۴ بایت داریم پس برای نشان دادن بایت ها به $2^4 = 16$ بیت نیاز داریم.

در هر بایت نیز ۴ کلمه داریم پس ۲ بیت نیاز داریم تا آدرس کلمه را مشخص کنیم.



۱۶ بیت

۲ بیت

۲ بیت



۱۸ بیت

آدرس کلمه

Index به معنای حجم کل حافظه نشان می‌دهد که برابر با $Line + offset$ می‌باشد که دارای ۴ بیت است.

★ **تدقیقات تکمیلی:** در این گونه سوالات معمولاً باید در ابتدا تعداد بیت های کل حافظه را پیدا کنیم پس با توجه به اعدادات سوال باید بقیه فیلدها را مشخص کنیم. فیلد $offset/word$ برابر با تعداد بیت های مورد نیاز برای غایش کلمه هست که یعنی: تعداد کلمات در فیلد $Line/Block$ برابر با تعداد بیت های مورد نیاز برای غایش تعداد بلوک ها یعنی: تعداد بیت ها در فیلد Tag برابر با $Line - offset$ - کل بیت ها

$Tag = 14 \text{ bits}$ $Index = 4 \text{ bits}$ $Block = 2 \text{ bits}$ $word = 2 \text{ bits}$

آدرس ۱۷۰ وقتی ارسال می شود، آدرس های ۱۶۸، ۱۶۹، ۱۷۰ و ۱۷۱ نیز به بلوک ۲ نگاشته می شوند.

بلوک ۲:

w_0	۱۶۸
w_1	۱۶۹
w_2	۱۷۰
w_3	۱۷۱

اگر hit رخ داده کاری نمی کنیم ولی اگر miss رخ دارد، اگر بلوک مورد نظر خالی بوده ۴ آدرس مساوی را جای گذاری می کنیم و اگر قبلاً اشغال شده باشد، هر ۴ آدرس قبلی پاک می شود و ۴ آدرس جدید جای آنها می سند.

به همین ترتیب ادامه می دهیم:

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

$$\text{Hit Rate} = \frac{21}{27} = 77\%$$

۲

AA: A 1010 \rightsquigarrow B_rW_r: A8, A9, AA, AB miss

101: 10 0001 \rightsquigarrow B_oW₁: 100, 101, 102, 103 miss

A8: A 1000 \rightsquigarrow B_rW₀ hit

103: 10 0011 \rightsquigarrow B_oW₃ hit

B0: B 0000 \rightsquigarrow B_oW₀: B0, B1, B2, B3 miss

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

۳) بابتجه به شکل مشاهده می شود که نرخ برخورد برنامه α بیشتر از ρ است که نشان دهنده این است

که میزان دسترسی به داده ها α محلی تر از ρ است. در برنامه C هم دسترسی به داده ها تقریباً تعدادی است.

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

$$t_{\text{ضرایب } j} = (1-h_1)(1-h_2)(1-h_3) \dots (1-h_{j-1}) = \prod_{i=1}^{j-1} (1-h_i) \quad (ب)$$

$j > 1$

ج) از آنجا که به ازای هر لایه $0 < h_i < 1$ ، بنابراین $0 < 1-h_i < 1$ پس ضرایب

t_1 از ضرایب هر n که $n > 1$ بیشتر است، پس کاهش t_1 بیشترین سهم کاهش T_{access} دارد

د) از آنجا که در صورت ثابت ماندن t_j ها، عبارت $(1-h_1)$ بیش از هر $(1-h_{j-1})$ (بسیار ضرایب $j-1$ ها ضرب شده است) پس افزایش h_1 یا کاهش $(1-h_1)$ بیشترین سهم را در کاهش T_{access} دارد.