حل تشریحی تست‌های معماری کامپیوتر سال ۹۸

تست ۷۳ سال ۹۸ ـ حداقل شرط لازم و کافی برای تشخیص دو عدد بی‌علامت A و B به طوری که $A \geq B$ باشد، با استفاده از $A + \bar{B} + 1$ کدام است؟

۱) $Z = 1 \; or \; S = 0$

۲) $Z = 1 \; or \; C = 1$

۳) $S = 0$

۴) $C = 1$

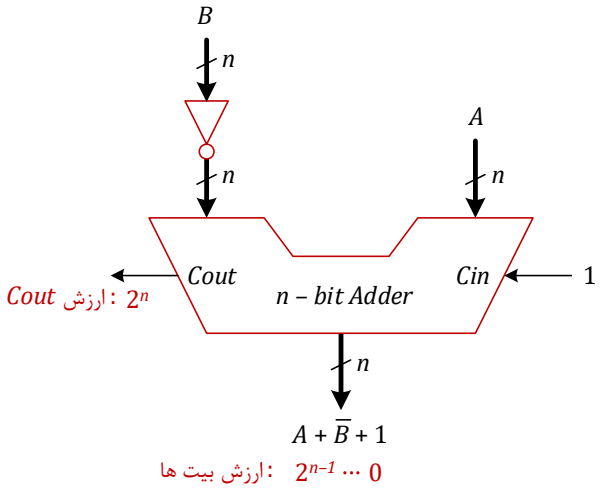
حل تست ۷۳ معماری کامپیوتر :

عبارت $(A)_2 + (\bar{B})_2 + 1$ در واقع جمع یک عدد با مکمل ۲ عدد دوم است، یعنی : $(A)_2 + [B]_2$:

طبق تعریف مکمل ۲ این عبارت برابر است با :

$$(A)_2 + [B]_2 = (A)_2 + (2^n - (B)_2) = 2^n + ((A)_2 - (B)_2)$$

که در آن n تعداد ارقام مجاز عددهای A , B است. دقت کنید که برای این عملیات از مدار زیر استفاده می‌شود:



شکل ۱ـ مدار جمع کننده مورد استفاده برای تفریق به روش مکمل ۲

همانگونه که مشخص است، خروجی رقم نقلی $Cout$ دارای ارزش 2^n است. حال چنانچه $(A)_2 \geq (B)_2$ باشد:

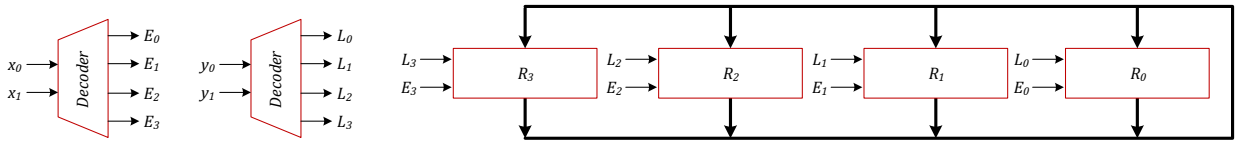
$$(A)_2 + [B]_2 = (A)_2 + (2^n - (B)_2) = 2^n + ((A)_2 - (B)_2) \geq 2^n$$

یعنی حاصل جمع از 2^n بزرگتر است و این بدین معنی که رقم نقلی 1 می‌شود و اگر $(A)_2 < (B)_2$ باشد:

$$(A)_2 + [B]_2 = (A)_2 + (2^n - (B)_2) = 2^n + ((A)_2 - (B)_2) < 2^n$$

یعنی حاصل جمع از 2^n بزرگتر است و این بدین معنی که رقم نقلی 0 خواهد شد. پس **گزینه ۴** صحیح است.

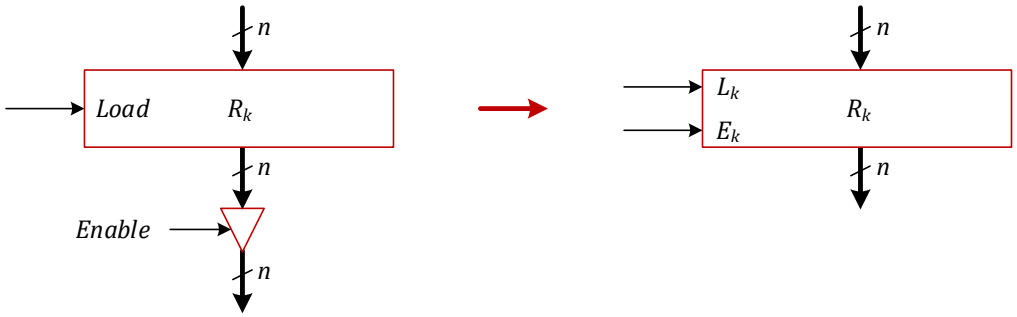
تست ۷۲ سال ۹۸ ـ در شکل زیر L_i فرمان Load و E_i فرمان Enable خروجی سه حالته ثبات i است. برای اینکه انتقال $R_3 \leftarrow R_2$ انجام شود، کدام کد عملیات باید به این مدار اعمال شود؟ (کد به Hex نشان داده شده و با فرمت $y_1\ y_0\ x_1\ x_0$ است.



۱) B
۲) C
۳) D
۴) E

حل تست ۷۲ معماری کامپیوتر :

برای آنکه ریزعمل $R_3 \leftarrow R_2$ انجام شود، لازم است ابتدا رجیستر R_2 به روی باس منتقل شود، بنابراین پایه Enable این رجیستر باید فعال شود ($L_2 = 1$). دقت کنید که پایه Enable در واقع بر اساس شکل زیر قابل توجیه است:



و از طرف دیگر پایه Load رجیستر R_3 باید یک باشد ($L_3 = 1$).

در نتیجه لازم است $x_1x_0 = 10$ و $y_1y_0 = 11$ شود. بنابراین $y_1y_0x_1x_0 = 1110 = E$ خواهد شد. **(گزینه ۴)**

تست ۷۱ سال ۹۸ ــ در یک کامپیوتر مجموعه دستورالعمل‌ها به گونه‌ای تغییر کرده است که ۲۰ درصد زمان یک برنامه ۴ برابر تسریع شده، ۳۰ درصد آن ۲ برابر کاهش سرعت یافته و نیز ۱۰ درصد از دستورات حذف شده است. سرعت این برنامه در حالت جدید نسبت به حالت قبل چه تغییر ی کرده است؟

۱) سرعت بدون تغییر است.
۲) سرعت افزایش یافته است.
۳) سرعت کاهش یافته است.
۴) مفروضات برای پاسخ کافی نیست.

حل تست ۷۱ معماری کامپیوتر : برای حل این مساله از تعمیم قانون آمدال (Amdahl's Law) استفاده می‌کنیم:

$$S = \frac{1}{\frac{f_1}{K_1} + \frac{f_2}{K_2} + \cdots + \frac{f_m}{K_m}}$$

این فرمول نشان‌دهنده میزان افزایش سرعت عملیات است زمانی که برنامه‌ای شامل m قسمت مختلف باشد که زمان اجرای هر کدام f_j ($j = 1,2, \ldots, m$) درصد از کل زمان اجرای برنامه باشد و این بخش را K_j برابر سریعتر از حالت اولیه اجرای کنیم.

بنابراین با جایگذاری مقادیر داده شده در این فرمول میزان افزایش سرعت را می‌توان محاسبه کرد.(دقت کنید بخشی از برنامه که حذف می‌شود را می‌توان تصور کرد که $f_j = 0$ شده است یا $K_j = \infty$). پس خواهیم داشت:

$$f_1 = \frac{20}{100} = 0.2 \quad , \quad K_1 = 4$$
$$f_2 = \frac{30}{100} = 0.3 \quad , \quad K_2 = \frac{1}{2}$$
$$f_3 = \frac{10}{100} = 0.1 \quad , \quad K_3 = \infty$$
$$f_1 = \frac{40}{100} = 0.4 \quad , \quad K_4 = 1$$

دو نکته دیگر در این جایگذاری دیده می‌شود، یکی اینکه اگر بخشی از برنامه به میزان h برابر کندتر اجرا می‌شود، می‌توان تصور کرد که به مقدار $K = \frac{1}{h}$ برابر سریعتر اجرا می‌شود. دیگر اینکه اگر بخشی از برنامه تغییری نکند، برای آن $K = 1$ در نظر گرفته می‌شود، اکنون به راحتی میزان افزایش سرعت برابر است با :

$$S = \frac{1}{\frac{0.2}{4} + \frac{0.3}{0.5} + \frac{0.1}{\infty} + \frac{0.4}{1}} = \frac{1}{1.05} < 1$$

پس کاهش سرعت وجود خواهد داشت. **(گزینه ۳)**

تست ۷۰ سال ۹۸ ـ دو عدد A=1010010 : مضروب و B=1110011 : مضروب فیه به روش Add & Shift در هم ضرب می‌شوند. تعداد عملیات جمع کدام است؟

۵) ۵ ۶) ۴
۷) ۳ ۸) ۲

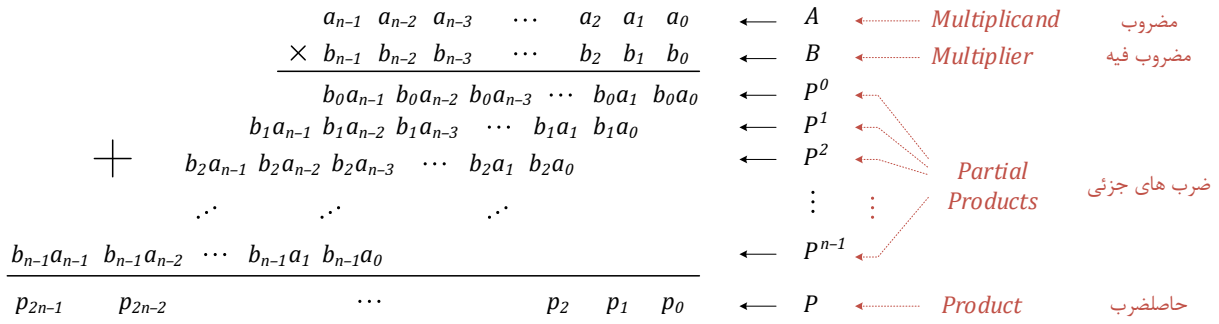
حل تست :

این تست را در پنجشنبه دو هفته قبل حل کرده‌ام، اما برای کامل بودن این مجموعه همان را مجدداً در اینجا ذکر می‌کنم.

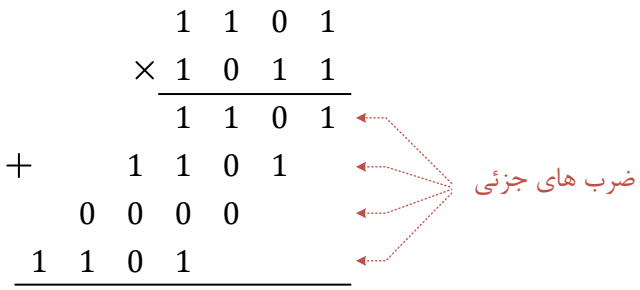
خلاصه مساله این است که تعداد عملیات جمع برابر است با تعداد یک های موجود در مضروب فیه. بنابراین برای انجام عمل ضرب به ۵ عمل جمع نیاز است. بنابراین **گزینه ۱** صحیح است.

توضیح تشریحی

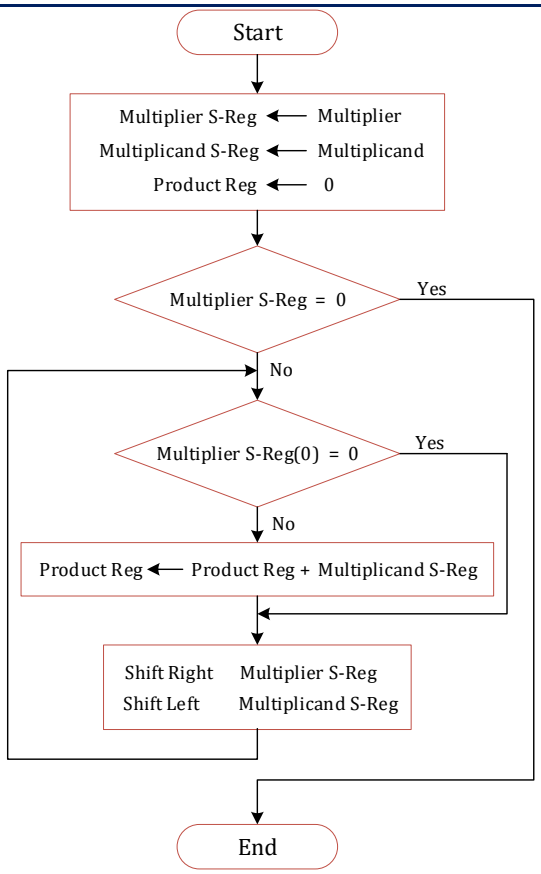
برای ضرب عدد A : مضروب (Multiplicand) در عدد B : مضروب فیه (Multiplier) عملیات ضرب به صورت زیر است:



در اینجا لازم است تا هر بیت از عدد B در عدد A ضرب شود و ضرب‌های جزئی (Partial Product) به دست آمده باهم جمع شوند. بدیهی است که اگر یک بیت از عدد B صفر باشد، ضرب جزئی صفر است و دیگر نیازی به جمع کردن آن نیست. به مثال زیر توجه کنید.



الگوریتم عمل ضرب به روش *Add & Shift* به صورت زیر است:



در این الگوریتم، ابتدا مضروب و مضروب فیه در دو شیفت رجیستر قرار می‌گیرند، و رجیستری به نام *Product* که نتیجه ضرب (حاصلضرب) در آن قرار خواهد گرفت صفر می‌شود. در مراحل بعد بیت‌های مضروب فیه چک می‌شوند، اگر صفر باشد، حاصلضرب جزئی صفر است، بنابراین نیازی به جمع کردن وجود ندارد، اما اگر صفر نباشد، حاصلضرب جزئی به دست آمده با *Product* جمع می‌شود و ... .

توجه کنید که اگر حاصلضرب جزئی غیرصفر باشد، عمل جمع انجام خواهد شد. بنابراین تعداد عملیات جمع برابر است با تعداد حاصلضرب‌های جزئی غیرصفر و این به معنی تعداد یک‌های موجود در مضروب فیه است.

اشتباهی که گاهی رخ می‌دهد این است که اگر قرار بود این عملیات با قلم وکاغذ انجام شود، به نظر تعداد عملیات جمع مورد نیاز از تعداد یک‌های موجود در مضروب فیه یکی کمتر خواهد بود. اما باید این نکته را در نظر گرفت که عملیات ضرب توسط سخت‌افزار انجام می‌شود و الگوریتم مورد اشاره در بالا اساس کار ضرب به روش *Add & Shift* است، نه روش‌های ذهنی.

موفق باشید