تمرین پنجم معماری کامپیوتر

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف

آرین احدی نیا

شماره دانشجویی: [redacted]

استاد درس: جناب آقای دکتر جهانگیر

دستیار آموزشی: جناب آقای علیپور

پاییز ۱۴۰۰

فهرست عناوین

طراحی واحد ALU

سوال ۱

اگر قلب این ALU را یک Adder با ورودی Carry در نظر بگیریم، اگر ورودی‌های این Adder را به صورت زیر در نظر بگیریم، خروجی‌های مورد نظر تولید می‌شود.

Function	In1	In2	Carry In
000	A	A	0
001	A	A	1
010	B	B	0
011	B	B	1
100	A	B	0
101	A	B	1
110	A	~B	0
111	A	~B	1

توجه کنید که با توجه به روش مکمل دوم داریم:

$$A - B = A + \sim B + 1$$

$$A - B - 1 = A + \sim B$$

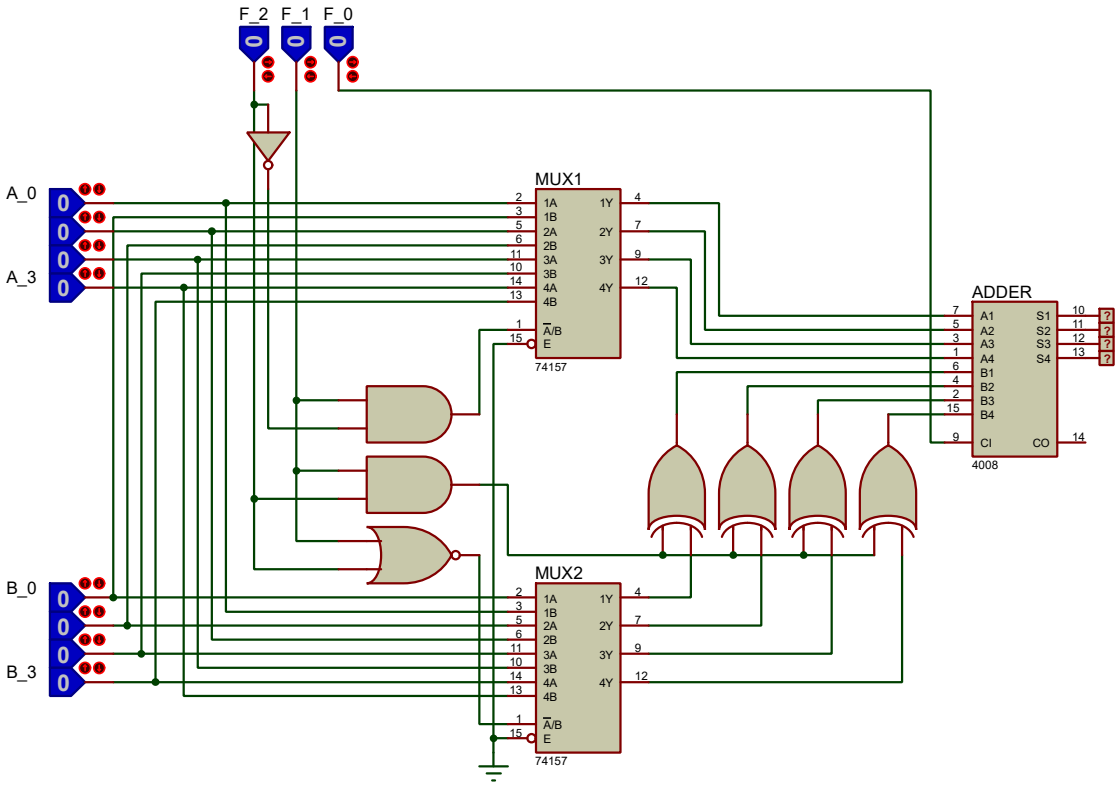
برای دادن ورودی به این پردازنده، هر ورودی را به یک MUX متصل می‌کنیم که هر یک به دو ورودی A و B متصل هستند. همچنین بین MUX و ورودی دوم ALU، یک گیت XOR قرار می‌دهیم که در صورت لزوم بتوانیم نقیض ورودی دوم را وارد ALU کنیم. توجه کنید که XOR مانند یک گیت NOT با ورودی active خواهد بود. با توجه به جدول فوق می‌توانیم ورودی‌های ALU را معین کنیم.

در MUX اول، اگر ورودی اول را به A و ورودی دوم را به B وصل کنیم، ورودی انتخابی به صورت $AND(F_2', F_1)$ خواهد بود. در MUX دوم، اگر ورودی اول را به B و ورودی دوم را به A وصل کنیم، ورودی انتخابی به صورت $NOR(F_2, F_1)$ قرار می‌دهیم. همچنین ورودی دیگر XOR را به صورت $AND(F_2, F_1)$ می‌توانیم در نظر بگیریم تا برای دو دستور آخر، نقیض B وارد ALU شود.

برای تکمیل مدار، کافی است که F_3 را به ورودی Carry In متصل کنیم. چرا که این دو سیگنال دقیقا مشابه یکدیگر هستند.

در نهایت با استفاده از نرم‌افزار Proteus می‌توانیم مدار فوق را بسازیم. ورودی A و B را در مدار زیر ۴ بیتی در نظر میگیریم چرا که مدارهای از پیش‌آماده در این نرم‌افزار ۴ بیتی هستند. توجه کنید که میتوانیم با متصل کردن زمین به ورودی پرارزش A و B، این مدار را به یک ALU سه‌بیتی تبدیل کنیم.

ادامه در صفحه بعد ...


F_2
F_1
F_0
A_0
A_3
B_0
B_3
MUX1
MUX2
74157
ADDER
4008
1A
1B
2A
2B
3A
3B
4A
4B
1Y
2Y
3Y
4Y
A/B
E
A1
A2
A3
A4
B1
B2
B3
B4
S1
S2
S3
S4
CI
CO

سوال ۲

ایده کار این مدار به این صورت است که هر یک از دو ورودی را به بخش‌های m بیتی تقسیم میکند، سپس هر یک از بخش‌های m بیتی، به جز بخش کم‌ارزش، را یک بار بدون ورودی carry و بار دیگر با ورودی carry جمع می‌کند. توجه کنید که تمام جمع‌ها به وسیله یک CRA با اندازه m بیت انجام میشود بنابرین حاصل‌ها همزمان با یکدیگر محاسبه می‌گردند.

اگر بخش‌های m بیتی را به ترتیب کم‌ارزش به پرارزش، بخش اول، دوم، سوم و ... بنامیم، میتوانیم با خروجی carry بخش اول، در بخش دوم انتخاب کنیم که آیا باید از حاصل محاسبه شده با carry یا بدون carry استفاده کنیم. به همین روند با توجه به خروجی carry بخش دوم، میتوانیم خروجی بخش سوم را مشخص کنیم و ...

توجه کنید که هر CRA از m عدد Full Adder تشکیل شده است. اگر تاخیر هر F.A. برابر t_{FA} باشد، تاخیر CRA برابر mt_{FA} خواهد بود.

اگر از $n-1$ بلوک که شامل دو CRA و یک MUX میشود برای جمع استفاده کنیم، و تاخیر MUX برابر t_{MUX} باشد، تاخیر نهایی برابر خواهد بود با

$$\mathrm{T} = t_{CRA} + (n-1)t_{MUX} = mt_{FA} + (n-1)t_{MUX}$$

توجه کنید که داریم.

$$nm = 100$$

بنابرین تاخیر نهایی برابر خواهد بود با

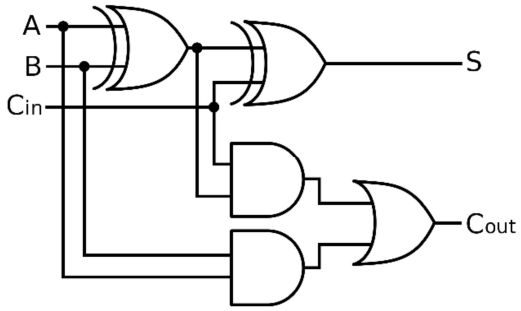
$$\mathrm{T} = t_{CRA} + (n-1)t_{MUX} = mt_{FA} + \left(\frac{100}{m} - 1\right)t_{MUX}$$

به شکل زیر میتوانیم تابع مورد نظر را کمینه کنیم.

$$\frac{\partial \mathrm{T}}{\partial m} = t_{FA} + (-\frac{100}{m^2})t_{MUX}$$

$$\frac{\partial \mathrm{T}}{\partial m} = 0 \Rightarrow \frac{t_{FA}}{t_{MUX}} = \frac{100}{m^2} \Rightarrow \left(\frac{m}{10}\right)^2 = \frac{t_{MUX}}{t_{FA}} \Rightarrow m = 10\sqrt{\frac{t_{MUX}}{t_{FA}}}$$

سوال ۳



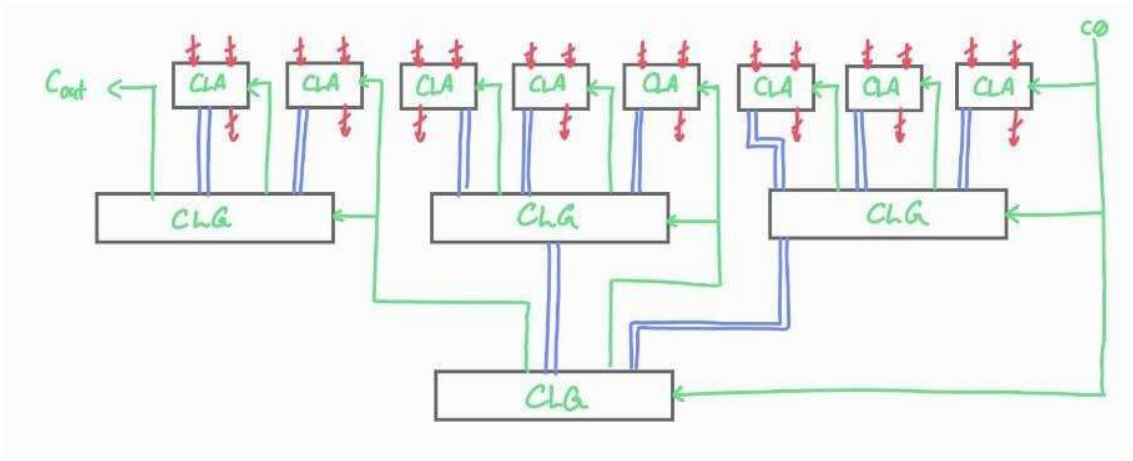
در تصویر فوق، شمای مدار یک Full Adder را مشاهده می‌فرمایید. از آنجایی که تاخیر گیت‌های پایه برابر 1Δ است. تاخیر تولید خروجی S و carry به ترتیب برابر 2Δ و 3Δ خواهد بود.

بنابرین برای محاسبه kامین رقم که از طریق CRA بدست می‌آید $3(k-1)+2)\Delta$ و برای محاسبه kامین رقم carry خروجی $3k\Delta$ زمان لازم است.

بنابرین برای محاسبه خروجی C_{31} که پرارزش‌ترین رقم خروجی است و C_{32} که آخرین رقم carry است، به ترتیب 95Δ و 96Δ زمان لازم است.

توجه بفرمایید که مدار FA را به صورت SOP نیز میتوان پیاده‌سازی کرد اما آنجا نیز گیت NOT باعث میگردد تا تاخیر با سه طبقه تاخیر محاسبه شود. بنابرین نیازی به ساده‌سازی FA نیست.

شکل زیر شمایی از مدار را CLA/CLG را نشان می‌دهد.



توجه بفرمایید که خروجی p و g واحد CLA به صورت زیر محاسبه می‌گردد.

$$p = p_3p_2p_1p_0$$

$$g = g_3 + p_3g_2 + p_3p_2g_1 + p_3p_2p_1g_0$$

با یک لایه، مقادیر p_i و g_i درونی محاسبه می‌گردد. پس از آن با تاخیر یک لایه خروجی p و با دو لایه خروجی g محاسبه می‌گردد. بنابرین مجموع تاخیر در محاسبه p و g برابر 3Δ است. بنابرین پس از سه واحد زمانی، همه CLAها خروجی p و g خود را آماده می‌کنند.

در CLG، خروجی‌ها به شکل زیر محاسبه می‌گردند.

$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

$$g = g_2 + p_2 g_1 + p_2 p_1 g_0$$

$$p = p_2 p_1 p_0$$

بنابرین در اینجا نیز خروجی‌ها با دو لایه تاخیر آماده می‌شوند. البته p با یک لایه تاخیر آماده می‌شود ولی به دلیل اینکه حضور g نیز نیاز است، تاخیر در آماده شدن p و g برابر 2Δ است.

بنابرین پس از آماده شدن ورودی p و g در CLAها لایه اول پس از 3Δ، پس از 2Δ خروجی‌های p و g لایه اول CLG آماده می‌شود و پس از 2Δ دیگر زمان، ورودی $carry$ لایه اول CLG آماده می‌گردد و پس از 2Δ ورودی carry برای CLAها آماده میشود. در این زمان خروجی C_{out} آماده است و پس از محاسبه $xor(p, c)$، نتیجه بیت‌ها نیز حاضر می‌شود.

بنابرین برای محاسبه خروجی C_{31} که پرارزش‌ترین رقم خروجی است و C_{32} که آخرین رقم carry است، به ترتیب 10Δ و 9Δ زمان لازم است.

همانگونه که ملاحظه فرمودید، در حالت دوم، ابتدا carry محاسبه شد و بر مبنای آن جمع صورت گرفت اما در حالت اول با محاسبه جمع، carry مرحله محاسبه می‌گشت.

سوال ۴

میتوانیم صفر بودن خروجی ALU را با NOR کردن تمام بیت‌های خروجی متوجه شویم. اگر حاصل یک باشد به این معنی است که تمام بیت‌های خروجی برابر صفر است و اگر صفر باشد به این معنی است که حداقل یکی از بیت‌های خروجی مخالف صفر است.

توجه کنید که خروجی جمع کننده دو عدد n بیتی یک عدد $n+1$ بیتی است. بنابرین با وصل کردن پرارزش‌ترین بیت به carry، میتوانیم وقوع یا عدم وقوع carry را معین کنیم.

اگر از سیستم نمایش مکمل دوم یا علامت ـ مقدار استفاده کنیم، با وصل کردن بیت علامت به Sign میتوانیم متوجه مقدار Sign شویم. در سیستم‌های نمایش دیگر نیز حسب قرارداد این کار با یک مدار ترکیبی امکان‌پذیر است.

تشخیص Overflow قدری پیچیده‌تر است. فرض می‌کنیم که از سیستم نمایش اعداد مکمل دوم استفاده می‌کنیم. همانگونه که مستحضر هستید، در این نمایش تفریق به جمع تبدیل می‌گردد. توجه بفرمایید که در جمع دو عدد با علامت‌های مخالف هیچگاه سرریز رخ نمیدهد چرا که اندازه حاصل قطعا از اندازه هر یک از اعداد کوچک‌تر است. تنها حالت روی دادن سرریز زمانی است که جمع برروی دو عدد هم‌علامت اجرا شود. توجه بفرمایید که سرریز تنها در Sign نمایان میگردد. بقیه ارقام به صورت طبیعی با یکدیگر جمع می‌شوند. در جمع رقم علامت، ۸ حالت وجود دارد. این ۸ حالت را میتوانید در جدول زیر ملاحظه فرمایید.

a	b	c_{in}	s	c_{out}	$overflow$	$c_{in} \oplus c_{out}$
0	0	0	0	0	-	0
0	0	1	1	0	+	1
0	1	0	1	0	-	0
0	1	1	0	1	-	0
1	0	0	1	0	-	0
1	0	1	0	1	-	0
1	1	0	0	1	+	1
1	1	1	1	1	-	0

همانگونه که ملاحظه می‌فرمایید، سرریز دقیقا زمانی رخ می‌دهد که در محاسبه بیت علامت (پرارزش)، $c_{in} \oplus c_{out}$ مقدار فعال داشته باشد. بنابرین با محاسبه این مقدار و وصل کردن آن به پرچم overflow میتوانیم مقدار این پرچم را محاسبه کنیم.

سوال ۵

توجه بفرمایید که Excess3 به صورت زیر است.

-3	0000
-2	0001
-1	0010
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100
10	1101
11	1110
12	1111

همانگونه که ملاحظه می‌فرمایید، کد Excess 3، سه واحد از معادل باینری خود بیشتر است.

$$Ex3(x) = x + 3$$

بنابرین برای جمع دو عدد $Ex(3)$، داریم

$$Ex3(x + y) = x + y + 3 = Ex3(x) + Ex3(y) - 3$$

توجه فرمایید که تا زمانی که $Ex3(x) + Ex3(y)$ کوچک‌تر از ۱۲ باشد، با کم کردن ۳ واحد به بازه نرمال برمی‌گردد. بنابرین تا زمانی که در جمع چهار بیتی سرریز رخ ندهد، در کل جمع نیز سرریز رخ نداده است. اما اگر حاصل جمع چهار بیتی سرریز کند، باید ۶ واحد به حاصل جمع اضافه کنیم تا حاصل از ۶ ترکیب مربوط به اعداد خارج از بازه عبور کند و به بازه مطلوب برسد. بنابرین حاصل جمع در این حالت را میتوانیم به صورت زیر در نظر بگیریم.

$$Ex3(x + y) = \begin{cases} Ex3(x) + Ex3(y) - 3 & on\ normal \\ Ex3(x) + Ex3(y) + 3 & on\ overflow \end{cases}$$

برای تفریق Excess 3 نیز به طریق مشابه میتوانیم عمل کنیم. توجه فرمایید.

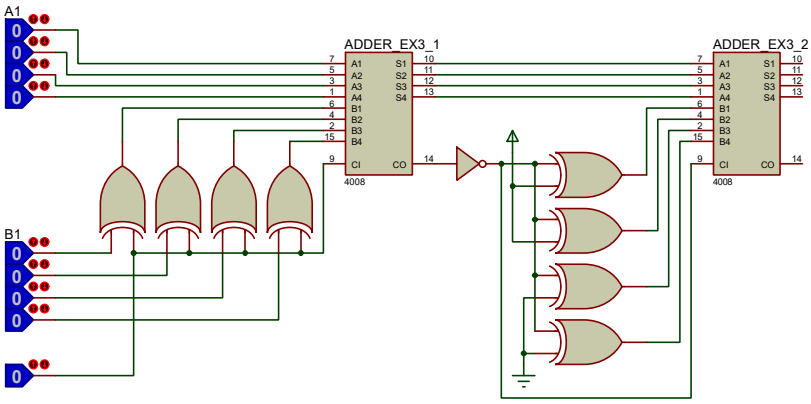
$$\sim Ex3(x) = 12 - x$$

$$Ex3(x - y) = Ex3(x) - 3 + \sim Ex3(y) - 12 + 3 = Ex3(x) + \sim Ex3(y) - 12$$
$$= Ex3(x) + (\sim Ex3(y) + 1) + 3$$

توجه بفرمایید که چون از روش مکمل دوم استفاده میکنیم، حاصل در حقیقت برابر $16 + x - y$ خواهد بود. بنابرین borrow رخ میدهد اگر و تنها اگر در جمع چهار بیتی $Ex3(x) + (\sim Ex3(y) + 1)$ سرریز رخ ندهد. بنابرین در صورتی که سرریز نداشته باشیم، باید از حاصل ۶ واحد کم کنیم تا به بازه نرمال برگردد.

$$Ex3(x+y)=\begin{cases}Ex3(x)+(\sim Ex3(y)+1)-3 & on\ normal\\ Ex3(x)+(\sim Ex3(y)+1)+3 & on\ overflow\end{cases}$$

مدار مربوط به جمع و تفریق کننده Excess 3 را میتوانیم به شکل زیر بسازیم.



حال جمع و تفریق BCD را بررسی می‌کنیم. در جمع BCD، اگر عدد بزرگتر از ۱۰ شود یا سرریز در جمع چهار بیتی اتفاق بیفتد، در کل جمع سرریز روی داده است و باید حاصل را با ۶ جمع کنیم تا به بازه نرمال برگردیم.

$$BCD(x+y)=\begin{cases}BCD(x)+BCD(y) & on\ normal\\ BCD(x)+BCD(y)+6 & on\ overflow\end{cases}$$

وقوع یا عدم وقوع overflow را می‌توانیم از طریق رابطه زیر کشف کنیم.

$$C_{BCD}=C_{out}+S_3S_2+S_3S_1$$

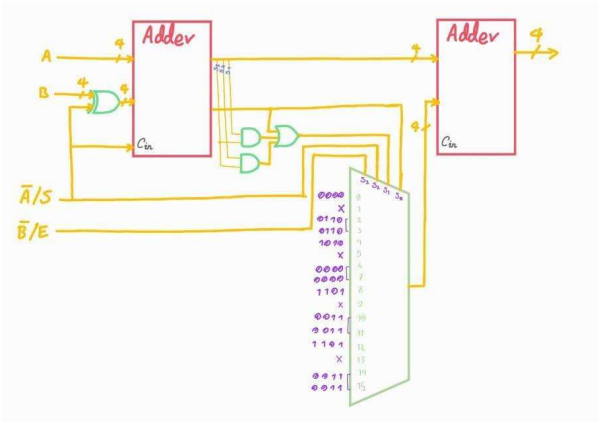
حاصل $BCD(x)+\sim BCD(y)+1$ را که همان $16+x-y$ است را در نظر بگیرید. این حاصل همان تفریق به‌کمک مکمل دو است. اگر $x\geq y$ باشد، حاصل بزرگتر از ۱۶ میشود و overflow رخ میدهد. در این حالت بیت قرضی ندارد. برعکس اگر $x<y$ باشد، حاصل کوچک تر از ۱۶ میشود و overflow رخ نمی‌دهد. در این حالت بیت قرضی داریم و از حاصل باید ۶ واحد کم کنیم تا به حالت نرمال برسد. بنابرین داریم.

$$BCD(x-y)=\begin{cases}BCD(x)+\sim BCD(y)+1-6 & on\ normal\\ BCD(x)+\sim BCD(y)+1 & on\ overflow\end{cases}$$

بنابرین هر ورودی که دریافت کنیم، ابتدا اولی را با دومی یا مکمل دوم دومی جمع میکنیم. سپس سیگنال های زیر مشخص میکنند که این عدد با چه عددی باید جمع شود. توجه کنید که سیگنال‌های مشخص کننده این امر، ۴ مورد هستند.

- $C_{EX3}=C_{out}$
- $C_{BCD}=C_{out}+S_3S_2+S_3S_1$
- $\overline{BCD}/EX3$
- $\overline{ADD}/SUB$

بنابرین برای تسریع هر چه بیشتر، این ۴ سیگنال را میتوانیم به یک MUX متصل کنیم و با توجه به آن، ورودی جمع‌کننده دوم را معین کنیم. توجه کنید که از ۱۶ حالت این MUX برخی از حالت‌ها بی‌اهمیت هستند. چرا که امکان پذیر نیست که C_{EX3} برابر ۱ باشد و C_{BCD} برابر صفر باشد. بنابرین در نهایت مدار را به شکل زیر می‌توانیم بسازیم


A
B
Adder
Adder
C_{in}
C_{in}
$\bar{A}/S$
$\bar{B}/E$

سوال ۶

تاخیر در انجام عملیات هیچ تغییری نمی‌کند. توجه بفرمایید که تنها همبستگی این سلول‌ها از طریق OR کردن دوتا از خروجی HA ها بدست می‌آید. دقیقا در همان زمانی که برای تولید ورودی نقلی از سلول اول به سلول دوم صرف میشود، سایر ورودی‌های نقلی نیز تولید می‌گردند. توجه بفرمایید که در اینجا منظور از نقلی، بیت carry در جمع نیست بلکه منظور بیتی است که بیت دو سلول جابجا می‌گردد.

ساختار خط‌لوله یا Pipeline

سوال ۱

(الف) توجه کنید که زمان هر سیکل در Pipeline به اندازه بیشینه زمان هر یک از Stageها خواهد بود.

اگر زمان اجرای هر یک از مراحل برابر t_i باشد و زمان اجرای Single Cycle برابر T باشد، خواهیم داشت

$$t_1 + t_2 + t_3 + \cdots + t_m \geq T$$

در نظر بگیرید که

$$t_{MAX} = MAX\{t_1, t_2, t_3, \dots, t_m\}$$

در این صورت طبق تعریف داریم

$$\forall i, t_{MAX} \geq t_i$$

بنابرین

$$mt_{MAX} \geq t_1 + t_2 + t_3 + \cdots + t_m \geq T$$

در نتیجه

$$t_{MAX} \geq \frac{T}{m}$$

توجه کنید که طبق آنچه پیشتر گفتیم، زمان هر سیکل در Pipeline به اندازه بیشینه زمان هر یک از Stageها خواهد بود. بنابرین اگر زمان سیکل در Pipeline برابر τ باشد، $\tau = t_{MAX}$ خواهد بود. بنابرین

$$\tau \geq \frac{T}{m}$$

در حالت آرمانی $\tau = \frac{T}{m}$ است. توجه کنید که رسیدن به این وضعیت آرمانی با قرار دادن $t_i = \frac{T}{m}$ امکان‌پذیر است.

توجه بفرمایید که اگر خط‌لوله شامل m مرحله شود، در $m-1$ مرحله اول، خط‌لوله در حال پر شدن است. بنابرین برای اجرای n دستورالعمل، به $n+m-1$ سیکل نیاز خواهیم داشت. بنابرین زمان اجرای n دستورالعمل برابر خواهد بود با $\tau(n+m-1)$. در صورتی که زمان اجرا برای Single Cycle برابر است با Tn. در این صورت میزان بهینه‌سازی برابر است با

$$\frac{Tn}{\tau(n+m-1)}$$

در حالت حدی که تعداد دستورالعمل ها زیاد باشد، این میزان افزایش سرعت برابر خواهد بود با

$$\lim_{n\to\infty} \frac{Tn}{\tau(n+m-1)} = \frac{T}{\tau}$$

از آنجایی که $\tau \geq \frac{T}{m}$ است، خواهیم داشت

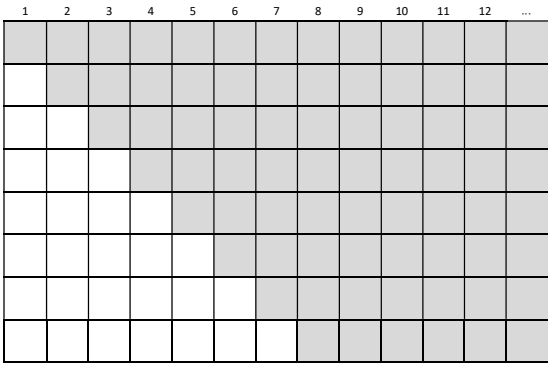
$$\lim_{n\to\infty} \frac{Tn}{\tau(n+m-1)} = \frac{T}{\tau} \leq m$$

بنابرین حداکثر میزان تسریع برابر m است. توجه کنید که این میزان تسریع در صورتی خواهد بود که هیچ پرشی کار خط‌لوله را مختل نکند.

(ب)

- در حد بینهایت، تسریع به تعداد طبقات میل می‌کند. اما اگر برنامه شامل پرش باشد، این تسریع به دست نخواهد آمد. توجه فرمایید که در صورت افزایش تعداد طبقات، پر شدن خط لوله در ابتدای کار و پس از پرش بیشتر طول خواهد کشید.
- در حالت معمول Pipelining، در حالت پرش، بین طبقات مختلف وابستگی ایجاد میگردد به نحوی که اجرای سایر دستورات، منوط به بروزرسانی PC خواهد شد. در صورتی که تعداد طبقات به طور دلخواه زیاد شود، این وابستگی ممکن است در موارد دیگر نیز بوجود آید و مشکلی مانند پرش ایجاد کند.
- در محاسبه رابطه فوق، از طریق ثبات واسط صرف نظر شده است. توجه بفرمایید که ثبات واسط نیز تاخیری را ایجاد می‌کند و آن مانع از رسیدن به تسریع مذکور می‌گردد.
- برای رسیدن به این تسریع، لازم است که بیشینه زمان اجرای کار طبقات برای $\frac{1}{m}$ باشد. شکاندن به طبقات متعدد از بابت محدودیت‌های معنایی که ایجاد میکند، این کار را غیر ممکن می‌سازد.

(ج) توجه کنید در صورتی که ۸ مرحله داشته باشیم، در ۷ سیکل اول، هنوز خط‌لوله به طور کامل پر نشده است.



در صورتی که هیچ پرشی کار خط‌لوله را مختل نکند و دستور از سیکل ۸ام به بعد به طور کامل در خط‌لوله قرار بگیرند و اجرا شوند، به ۱۰۰+۷ سیکل برای اجرای کامل این دستورات نیاز خواهیم داشت که برابر است با ۱۰۷ سیکل.

(د) این بخش را با یک مثال را توضیح می‌دهیم.

توجه کنید که سیکل خط‌لوله باید به اندازه بیشینه زمان اجرای بخش‌های مختلف آن باشد. در مواردی مانند ALU، زمان اجرا بسته به ورودی دارد. به عنوان مثال بدیهتا ضرب در ALU بسیار بیشتر از جمع زمان خواهد برد. در حالت عادی، باید بیشینه تاخیر ALU را در نظر بگیریم. اما یک تفکر هوشمندانه تر آن است که سیستم Pipeline را مجهز به سیگنال کنترلی Wait کنیم. در این صورت میتوانیم زمان سیکل‌ها را کوتاه‌تر کینم و در صورتی که در یک سیکل فرآیند در طبقات مختلف به

اتمام نرسید، سیگنال Wait فعال شود تا دستورات در همان وضعیت در خط‌لوله بمانند. از این جهت که بیشتر دستورالعمل‌ها از نوع جمع هستند. این روش می‌تواند به ما در سریع کردن فرآیند کمک کند. حال میتوانیم پیش از طبقاتی که گاهی زمان‌بر هستند، چند طبقه خالی قرار دهیم تا در صورت فعال شدن Wait، طبقات پیشین به کار خود برای آماده کردن دستورات بعدی ادامه دهند. توجه کنید که در روابط فوق نیز تعداد طبقات تنها در زمان لازم برای پر شدن خط‌لوله تاثیر گذار است و به در حالت پایدار زمان اجرای دستورات را افزایش نمی‌دهد. بنابرین اضافه کردن این طبقات با این استدلال می‌تواند موجب کاهش زمان اجرای دستورات شود.

سوال ۲

اگر هر طبقه با زمان T را به m بخش مساوی تقسیم کنیم، زمان هر بخش $\frac{T}{m}$ خواهد بود. اما هر یک از بخش ها زمان سربار $0.04mT$ خواهد داشت. بنابرین سیکل‌های پردازنده جدید به اندازه t طول خواهد کشید.

$$t(m,T) = \frac{T}{m} + 0.04mT$$

میخواهیم تابع فوق را نسبت به m کمینه کنیم، بنابرین مشتق تابع باید در آن نقطه برابر صفر باشد.

$$\frac{\partial t}{\partial m} = -\frac{T}{m^2} + 0.04T$$

$$\left.\frac{\partial t}{\partial m}\right|_{m_0} = 0 \Rightarrow -\frac{T}{(m_0)^2} + 0.04T = 0 \Rightarrow \frac{1}{(m_0)^2} = \frac{1}{5^2} \overset{m_0 \in \mathbb{N}}{\Longrightarrow} m_0 = 5$$

بنابرین به ازای $m = 5$ بیشترین بهینه سازی را خواهیم داشت. در این صورت زمان اجرای هر طبقه برابر خواهد شد با

$$t(5,T) = \frac{T}{5} + 0.2T = 0.4T$$

سوال ۳

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	...
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	...
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	...
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	...
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	...

در حالت بهینه Single Cycle، اگر کل فرآیند در زمانی معادل جمع زمان مرحله‌ها انجام شود، کل فرآیند در ۱۲ نانوثانیه انجام خواهد شد.

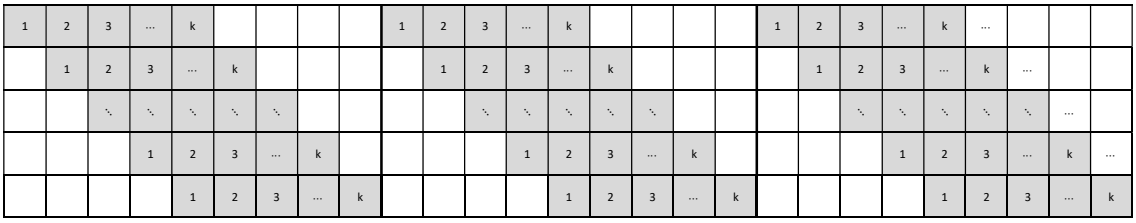
از طرف دیگر اگر این کار را به صورت Pipeline انجام دهیم، باید هر Cycle را به اندازه بیشینه زمان اجرای مراحل که برابر ۴ نانوثانیه است در نظر بگیریم. با صرف نظر از مراحل ابتدایی که Pipeline در حال پر شدن است و فرض اینکه پرشی انجام کار رو مختل نمی‌کند، در هر سیکل به طور متوسط یک دستورالعمل اجرا می‌شود. بنابرین زمان اجرا هر دستورالعمل به صورت حدی ۴ نانوثانیه خواهد بود. بنابرین تسریع این ساختار برابر خواهد بود با

$$o = \frac{T_{prev}}{T_{current}} = \frac{12ns}{4ns} = 3$$

سوال ۴

پاسخ این سوال تا حدی به نحوه رفتار خط‌لوله با دستورات پرش مربوط میشود.

فرض کنید که در هر k دستور، دستور آخر دستور پرش است و خط‌لوله پس از ورود دستور پرش تا زمان خروج ان جلوی ورود دستورات دیگر را میگیرد.



در این حالت همان‌گونه که مشاهده می‌فرمایید، اگر خط‌لوله m مرحله داشته باشد، در هر $k+m-1$ سیکل، k دستور اجرا میشود. بنابرین در این حالت افزایش تعداد دستورالعمل‌ها تاثیری ندارد. در این حالت بازده خط‌لوله برابر خواهد بود با

$$\frac{k}{k+m-1}$$

اما اگر رفتار خط‌لوله با پرش کمی هوشمندانه‌تر باشد، به عنوان مثال مانند سوال ۶، پس از گذر دستور پرش از مرحله pام از آخر، اجازه ورود دستورات بعدی داده شود، در این صورت اگر تعداد کل دستورالعمل‌ها kn باشد، به تعداد

$$kn+(m-p)(n-1)+m-1$$

سیکل برای اجرای کل دستورات نیاز است. در این حالت بازده برابر میشود با

$$\frac{kn}{kn+(m-p)(n-1)+m-1}$$

توجه بفرمایید که حالت $p=1$ دقیقا مشابه حالتی است که پیشتر توضیح دادیم.

اگر در نظر بگیریم که $i=kn$، در این صورت تعداد دستورالعمل‌ها برابر i خواهد بود. بنابرین بازده را میتوانیم به صورت زیر بازنویسی کنیم.

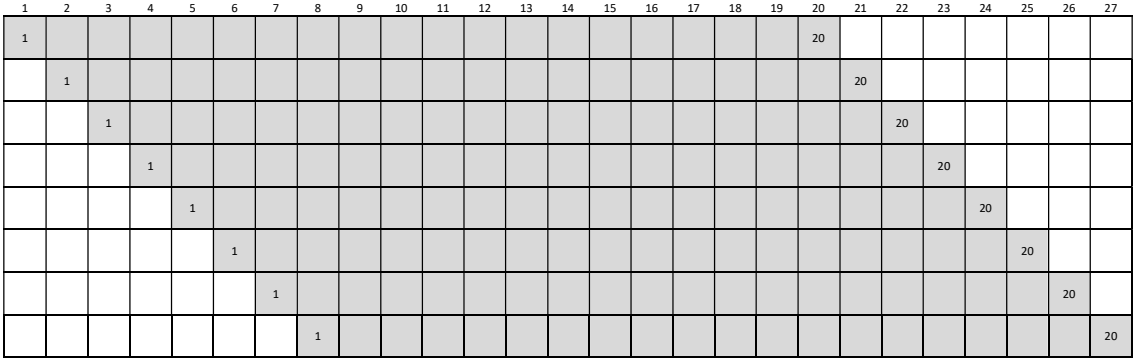
$$\frac{i}{i+(m-p)\left(\frac{i}{k}-1\right)+m-1}=\frac{i}{i\left(1+\frac{m-p}{k}\right)+p-1}$$

توجه بفرمایید که مقدار $1+\frac{m-p}{k}$ ثابت است. اگر آن را c در نظر بگیریم، خواهیم داشت

$$\frac{i}{i\left(1+\frac{m-p}{k}\right)+p-1}=\frac{i}{ic+p-1}=\frac{1}{c}\left(1-\frac{p-1}{ic+p-1}\right)$$

همانگونه که ملاحظه می‌فرمایید، این رابطه نسبت به i، تعداد دستورالعمل‌ها، صعودی است و در بینهایت از سمت پایین به $\frac{1}{c}$ میل میکند. بنابرین گزینه **ب** صحیح است.

سوال ۵

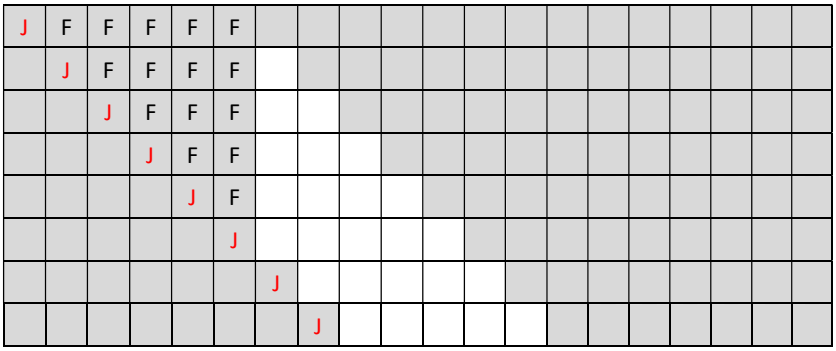


شکل فوق را خط‌لوله پردازنده در نظر بگیرید که هر سطر آن یک مرحله از خط‌لوله است. توجه بفرمایید که در ابتدای امر، خط‌لوله خالی است و نیاز است تا پس از گذشت چند سیکل، خط‌لوله پر شود. از همان ابتدا در هر سیکل یک دستور وارد خط‌لوله می‌شود تا اینکه در سیکل بیستم پس از گذشت ۱۹ سیکل، دستور پرش وارد خط‌لوله می‌شود و مانع ورود سایر دستورالعمل‌ها می‌گردد. پس از چند سیکل، این دستور به طور کامل اجرا می‌گردد و در نهایت پس از ۲۷ سیکل، این فرآیند که شامل اجرای ۲۰ دستورالعمل می‌شد، پایان می‌پذیرد. برنامه ما ۱۰۰ دستورالعمل دارد پس پنج بار فرآیند فوق باید تکرار شود.

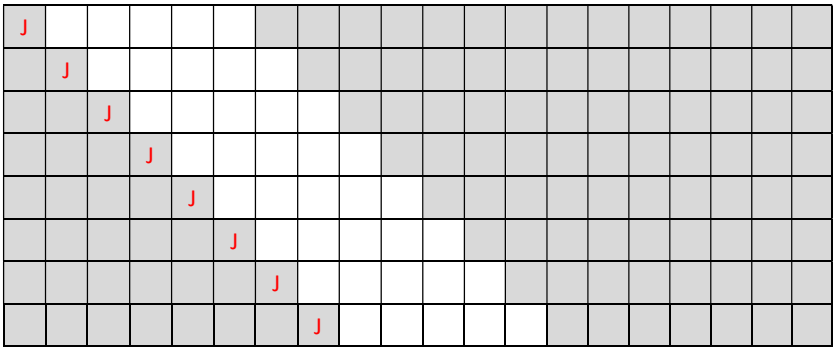
توجه کنید که هر مرحله از خط‌لوله در کل ۱۰ نانوثانیه است. بنابرین هر سیکل ۱۰ نانوثانیه خواهد بود. بنابرین زمان کل برای اجرای برنامه مورد نظر عبارت است از

$$5 \times 27 \times 10ns = 1350ns$$

سوال ۶



فرض کنید دستوری که با J مشخص شده است، دستوری است که به ازای آن پرش انجام می‌شود و دستوراتی که با F مشخص شده‌اند نیز دستوراتی هستند که به علت خالی شدن خط‌لوله در مرحله ششم، تخلیه می‌شوند. همانگونه که ملاحظه می‌فرمایید، ۵ دستور بعد از پرش تخلیه میشوند. بنابرین به نوعی می‌توانیم در نظر بگیریم که پردازنده در پنج سیکل پس از دستور پرش، کاری را انجام نداده است.



بنابرین اگر n دستور داشته باشیم که mتای آنها از نوع پرش است، با صرف نظر از ۷ مرحله اولیه که صرف پر شدن خط‌لوله می‌گردد، تعداد سیکل‌های لازم برای اجرای این دستورات برابر $n + 5m$ است. اگر هر سیکل τ واحد زمانی به طول بینجامد، در مجموع انجام پردازش $\tau(n + 5m)$ زمان خواهد برد.

در پردازنده تک سیکل، دیگر چنین مشکلی نخواهیم داشت، هر دستور در یک زمان مشخص اجرا خواهد شد. اگر زمان اجرا در هر سیکل T باشد. زمان لازم برای اجرای کل عملیات برابر $\mathrm{T}n$ خواهد بود.

بنابرین افزایش سرعت در این پردازنده برابر خواهد بود با

$$\frac{\mathrm{T}n}{\tau(n+5m)} = \frac{\mathrm{T}}{\tau} \times \frac{n}{n+5m}$$

حال می‌خواهیم که تعداد دستورات پرشی را بدست آوردیم. طبق احتمال‌های داده شده داریم.

$$P(J) = P(J|UCJ)P(UCJ) + P(J|CJ)P(CJ)$$

منظور از UCJ، پرش غیر شرطی و از CJ پرش شرطی است. توجه کنید که $P(J|UCJ)$ برابر ۱ است. بنابرین داریم

$$P(J) = 1 \times 0.06 + 0.8 \times 0.10 = 0.14$$

بنابرین احتمال اینکه یک دستورالعمل شرطی باشد، برابر ۰/۱۴ است. اگر n دستورالعمل داشته باشیم، به طور میانگین تعداد پرش‌ها برابر خواهد بود با

$$m = E[J_n] = 0.14n$$

بنابرین افزایش سرعت برابر می‌شود با

$$\frac{\mathrm{T}}{\tau} \times \frac{n}{n + (5 \times 0.14)n} = \frac{\mathrm{T}}{\tau} \times \frac{1}{1.7}$$

اگر فرض کنیم که $\mathrm{T} = 8\tau$ است، خواهیم داشت

$$\frac{\mathrm{T}}{\tau} \times \frac{1}{1.7} = \frac{8}{1.7} = 4.7$$

بنابرین حدودا ۴/۷ برابر افزایش سرعت خواهیم داشت. توجه کنید که محاسبه این افزایش سرعت با فرض ایده‌آل $\mathrm{T} = 8\tau$ انجام شده است.

سوال ۷

گروه دستورات	IF	ID	EX	EF	MA	WB	درصد استفاده
۱	×	×	×	×	×	×	۱۰٪
۲	×	×	×			×	۳۵٪
۳	×	×		×	×		۲۰٪
۴	×	×	×				۳۵٪
۵	×	×					۵٪
زمان اجرا	8ns	2ns	4ns	5ns	10ns	2ns	

زمان اجرای هر دستور را به تفکیک محاسبه میکنیم.

گروه دستورات	درصد استفاده	زمان اجرا
۱	۱۰٪	31ns
۲	۳۵٪	16ns
۳	۲۰٪	25ns
۴	۳۰٪	14ns
۵	۵٪	10ns

در حالت بهینه Single Cycle، هر دستور دقیقا در زمان مورد نیاز اجرا خواهد شد. بنابرین زمان تخمینی اجرای دستورات برابر خواهد بود با

$$E[t] = 18.4ns$$

در صورتی که از ساختار خط‌لوله استفاده کنیم و هر یک از مراحل فوق را به عنوان یک Stage در نظر بگیریم، یک خط‌لوله با ۶ عدد Stage خواهیم داشت. حداکثر زمان اجرا این Stage ها برابر 10ns است بنابرین هر سیکل از خط‌لوله 10ns زمان خواهد برد. البته توجه کنید که میتوانیم مرحله IF و ID را تجمیع کنیم تا یک مرحله 10ns بدست آید. همین کار را با EX و EF نیز میتوانیم انجام دهیم. در این صورت تعداد مراحل کم می‌شود اما زمان هر سیکل بدون تغییر می‌ماند. لذا زمان اجرا بهبود پیدا خواهد کرد.

اگر n دستورالعمل داشته باشیم، با فرض آرمانی اینکه هیچ پرشی که کار خط‌لوله را بهم بریزد در دستورات وجود ندارد، به $(4-1)+n$ سیکل برای انجام دستورالعمل‌ها احتیاج داریم. توجه کنید که آن ۳ سیکل مربوط به پر شدن خط‌لوله می‌شود.

بنابرین میزان تسریعی که از این خط‌لوله در حالت آرمانی دریافت می‌کنیم، برابر است با

$$\frac{n \times 18.4ns}{(3+n) \times 10ns}$$

که در حالت حدی این مقدار برابر می‌شود با

$$\lim_{n\to\infty} \frac{18.4}{(3+n) \times 10} = \frac{18.4}{10} = 18.4$$