به نام خدا

دانشکده مهندسی برق و کامپیوتر | مبانی کامپیوتر و برنامه‌نویسی

پاسخ تکلیف دوم – پاییز ۹۲ | تاریخ تحویل: ۲۷ مهر

پاسخ سوال ۱:

(a

11000001100110001100011000110001 = -17.1

(b

3F9FFFFF = 1.25

پاسخ سوال ۲:

جمع در قالب IEEE 754 انجام گرفته است پس هر سه عدد ۳۲ بیتی تا ٦٤ بیتی هستند. از آنجایی که عدد سمت راست تساوی ٦٤ بیتی است، اعداد سمت چپ نیز ۳۲ بیتی هستند لذا X چهار بیتی است همانند زیر است. هر حرف انگلیسی یک بیت را نشان می‌دهد:

A	B	C	D

برای بدست آوردن X کافی است معادل مبنای دو هر سه عدد را بدست آورده و با هم جمع کنیم. از روی جمع این اعداد چهار معادله بدست می‌آید که مقادیر A، B، C و D را مشخص می‌کند. از سمت چپ اعداد به صورت دودویی برابر هستند با (نقطه جدا کننده بخش اعشاری و صحیح است):

$$1.ABCD000000\ldots \times 2^{129-127} = 1AB.CD$$

$$1.ABCD0000\ldots \times 2^{130-127} = 1ABC.D$$

$$1.001011000\ldots \times 2^{131-127} = 10010.11$$

طبق صورت سوال رابطه زیر برقرار است:

$$1AB.CD + 1ABC.D = 10010.11$$

برای درک بهتر معادله بالا را به صورت زیر می‌نویسیم تا بیت‌های هم ارزش زیر هم قرار گیرند:

		1	A	B	.	C	D
+	1	A	B	C	.	D	0
1	0	0	1	0	.	1	1

حال معادله را به صورت دودویی حل می‌کنیم. دقت کنید که معادله باید از سمت راست به سمت چپ حل شود زیرا ممکن است بیت نقلی تولید شود که در حل معادله تاثیر گذار است.

نقلی
↓

$$D + 0 + 0 = 1 \Rightarrow D = 1$$
$$C + 1 + 0 = 1 \Rightarrow C = 0$$
$$B + 0 + 0 = 0 \Rightarrow B = 0$$
$$A + 0 + 0 = 1 \Rightarrow A = 1$$

پس $X = 1001$ است.

پاسخ سوال ۳:

	0010	0011
+	0011	0101
	0101	1000

از آنجایی که هیچیک از رقم‌های حاصل‌جمع از ۹ بزرگتر نشد، نیاز به تغییر نیست وگرنه لازم بود رقم‌های بزرگتر از ۹ با ٦ جمع شود تا عدد بدست آمده درست باشد.

پاسخ سوال ۴:

سطر اول خروجی را مشخص می‌کند و سطر دوم ورودی متناظر با هر کاراکتر.

H	e	l	l	o
72	101	108	108	111

i	t	p		9	2		@		s	b	u
105	116	112	32	57	50	32	64	32	115	98	117

برای تبدیل حروف کوچک به بزرگ کافی است از کد اسکی حروف کوچک ۳۲ واحد کم کنیم تا به کد اسکی حرف بزرگ معادل آن برسیم.

پاسخ سوال ۵:

ایرادها به تفکیک هر خط عبارتند از:

۱. I در Int نمی‌تواند بزرگ باشد و long کلمه کلیدی در زبان C است و نمی‌تواند به عنوان نام متغیر انتخاب شود.

۲. متغیر در مقداردهی باید در سمت چپ قرار گیرد.

۳. قرار دادن مقدار منفی در یک متغیر از نوع بی‌علامت، بی‌معنی است. اگرچه خطای دستوری نیست اما ممکن است منجر به خطای منطقی گردد زیرا انتظار می‌رود که درون متغیر مقدار ٤- باشد در حالیکه مقدار دیگری در آن قرار گرفته است.

۴. اگر هدف انتساب باشد، باید b را قبلاً تعریف کرده باشیم. اگر هدف مقداردهی باشد، باید نوع متغیر را تعیین کنیم.

۵. عمل انتساب به متغیر سمت چپ تساوی انجام می‌گیرد لذا برای ذخیره کردن نتیجه ضرب باید عبارت در سمت راست و متغیر مقصد در سمت چپ قرار گیرد. float کلمه کلیدی در زبان C است و نمی‌تواند به عنوان نام متغیر انتخاب شود. همین مسئله در مورد long نیز صادق است.

پاسخ سوال ۶:

مقداردهی اولیه یعنی تعیین مقدار یک متغیر در هنگام تعریف آن در حالیکه انتساب یعنی تعیین مقدار متغیر پس از تعریف آن. برای مثال در خط زیر متغیر number مقدار دهی اولیه می‌شود:

```
int number = 1;
```

ولی در خط زیر متغیر counter که قبلاً تعریف شده است، فرآیند انتساب بر روی آن انجام می‌گیرد:

```
counter = 100;
```

اگرچه در ظاهر این دو تفاوتی با هم ندارند و هر دو یک کار را انجام می‌دهند اما در آینده خواهید دید که زبان‌های C و C++ رفتار متفاوتی نسبت به این دو از خود نشان می‌دهد. مثلاً برخی متغیرها را می‌توان به سادگی مقداردهی اولیه کرد در حالیکه انتساب آن‌ها به ممکن نیست!

پاسخ سوال ۷:

الف)

```
#include <stdio.h>
int main()
{
	printf("Ali Ahmadi\n92213077");
	return 0;
}
```

راه حل‌های دیگری نیز وجود دارد. اما از آنجایی که نیاز به وارد کردن اطلاعات از طرف کاربر نیست، می‌توان به راحتی و تنها با یک printf اطلاعات خواسته شده را نمایش داد.

ب)

```
#include <stdio.h>
int main()
{
	printf("99.99%% of programming is your endeavor \\n \n:)\n");
	return 0;
}
```

دقت شود که علامت % و \ دو کاراکتر خاص هستند، که برای کار نمایش آن‌ها، باید به شیوه خاص عمل کرد. برای نمایش علامت % باید از %% و برای نمایش \ باید از \\ استفاده کرد.

پاسخ سوال ۸:

تنها خطوط ۱۰ تا ۱۵ از printf استفاده شده که مقداری را در خروجی نمایش می‌دهد. خروجی واقعی برنامه به صورت زیر است:

```
a = 4294967288
b = 97
c = W
d = -858993459
e = -3
f = 2
```

اما چرا این خروجی‌ها تولید شده است؟ دلیل آن در دو راز نهفته است. اولاً نحوه ذخیره‌سازی مقدار متغیر در حافظه و دوم این نکته که printf به شیوه‌ای که شما به آن گفته‌اید، به محتوای متغیر (قسمتی از حافظه) نگاه می‌کند. یعنی ممکن است در نگاه‌های مختلف به یک متغیر، مقادیر متفاوتی تولید شود. نحوه نگاه کردن همانی است که توسط علامت % تعیین می‌شود. برای مثال %d یعنی نگاه به متغیر به صورت عدد صحیح علامت‌دار (مکمل ۲)، %u یعنی نگاه به صورت عدد صحیح بدون علامت، %c یعنی نگاه به صورت

کاراکتر (یک بایتی) و %f یعنی نگاه به متغیر به صورت عدد اعشاری در قالب IEEE 754. پس برای درک پاسخ‌ها بهتر است ابتدا مقدار واقعی ذخیره شده در هر متغیر مستقل از نوع آن را بررسی کرده و سپس نحوه نگرش printf به آن را بررسی کنیم.

خطوط ٤ تا ٩ به مقدار دهی متغیرها می‌پردازد. تحلیل این خطوط عبارتند از:

١. در خط ٤ مقدار ٨- درون a قرار گرفته است. چون نوع int به معنی عدد صحیح علامت‌دار است نحوه ذخیره‌سازی ٨- به صورت مکمل دو خواهد بود. پس اگر در مبانی ١٦ به متغیر a نگاه کنیم، مقدار $FFFFFFF8_{16}$ درون آن قرار گرفته است (این عدد همان معادل مکمل ٢ عدد ٨- است).

٢. در خط ٥ مقدار 'a' درون متغیر b قرار گرفته است. مقدار واقعی درون b همان کد اسکی معادل با کاراکتر 'a' خواهد بود. این مقدار همان ٩٧ دهدهی یا 61_{16} می‌باشد.

٣. در خط ٦ مقدار ٨٧ مبنای ١٠ یا همان 00000057_{16} درون متغیر c قرار می‌گیرد.

٤. در خط ٧ مقدار ٢/١- در قالب IEEE 754 با دقت مضاعف درون d قرار می‌گیرد. پس از یک محاسبه ساده نتیجه می‌شود که مقدار $C000CCCCCCCCCCCD_{16}$ درون این متغیر قرار گرفته است. (برای تبدیل آنلاین اینجا را کلیک کنید)

٥. در خط ٨ عدد ٣- باید درون e قرار گیرد. اما e از نوع عدد صحیح بی‌علامت است ولی ٣- علامت‌دار است. در اینجا معادل مکمل ٢ عدد ٣- محاسبه شده و درون e قرار می‌گیرد. پس درون e مقدار $FFFFFFFD_{16}$ قرار دارد. دقت کنید که نباید بگوئیم عدد ٣- درون e ذخیره شده است، زیرا ٣- علامت‌دار است در حالیکه e بی‌علامت است؛ بلکه عدد واقعی ذخیره شده در e مقدار ٤٢٩٤٩٦٧٢٩٣ است.

٦. در خط ٩ یک عدد اعشاری می‌خواهد درون متغیری از نوع صحیح بدون علامت قرار گیرد. در این گونه موارد ابتدا قسمت اعشاری عدد حذف شده و مقدار صحیح درون متغیر قرار می‌گیرد. پس درون f معادل عدد ٢ در به صورت بدون علامت ذخیره می‌شود که برابر با 00000002_{16} است.

اکنون با توجه به نگرش printf به محتوای متغیرها، مقدار تولید شده را تحلیل می‌کنیم:

١. در خط ١٠ به متغیر a به صورت عدد صحیح بی‌علامت نگاه می‌شود. در بالا دیدیم که در a مقدار $FFFFFFF8_{16}$ ذخیره شده است. پس printf معادل بدون علامت این مقدار یعنی عدد ٤٢٩٤٩٦٧٢٨٨ را چاپ می‌کند.

٢. در خط ١١ از printf خواسته شده که به متغیر b به صورت علامت‌دار نگاه کند. دیدیم که در b مقدار 61_{16} ذخیره شده است. اما این مقدار یک بایتی است در حالیکه %d به معنی عدد علامت‌دار چهار بایتی است. بدین منظور مقدار یک بایتی باید به مقدار ٤ بایتی گسترش یابد. قانون گسترش بسیار ساده است. بیت پر ارزش باید به اندازه کافی تکرار شود. پس باید 61_{16} را گسترش دهیم که حاصل 00000061_{16} است. اکنون printf معادل مکمل ٢ مقدار محاسبه شده را نشان می‌دهد که برابر با ٩٧ است. این همان کد اسکی 'a' است.

۳. در خط ۱۲ از printf خواسته شده مقدار داخل c را با دید کاراکتر چاپ کند. نکته مهم آن است که c یک متغیر چهار بایتی است در حالیکه کاراکتر یک بایتی است. برای حل این مشکل، کامپایلر بیت‌های اضافی را حذف می‌کند. اینجا نیز یک قانون ساده وجود دارد که می‌گوید: «حذف از بیت‌های پرارزش صورت می‌گیرد». پس 00000057_{16} به 57_{16} تغییر یافته و کاراکتر معادل آن یعنی W (کد اسکی ۸۷) را چاپ می‌شود. دقت کنید که printf با مقدار درون c کاری ندارد!

٤. در خط ۱۳ به d به عنوان عدد صحیح علامت‌دار مثبت نگاه می‌شود. در بالا دیدیم در d مقدار $C000CCCCCCCCCCCD_{16}$ ذخیره شده است که یک مقدار ٦٤ بیتی است. لذا ابتدا باید به ۳۲ بیت تبدیل شود. طبق قانون، باید بیت‌های پر ارزش حذف شوند، لذا به عدد $CCCCCCCD_{16}$ می‌رسیم که معادل ۸۵۸۹۹۳٤٥۹– است. پس خروجی printf عدد ۸۵۸۹۹۳٤٥۹– است.

٥. در خط ۱٤ به مقدار داخل e به عنوان عدد علامت‌دار نگاه می‌شود. در بالا گفتیم که در e مقدار $FFFFFFFD_{16}$ ذخیره شده است. معادل مکمل ۲ این مقدار برابر با ۳– است. لذا خروجی ۳– خواهد بود.

٦. در خط ۱٥ از printf خواسته‌ایم به f به عنوان عدد صحیح بدون علامت نگاه کند. در بالا گفتیم که در f مقدار 00000002_{16} ذخیره شده است که معادل ۲ بدون علامت است. پس خروجی ۲ خواهد بود.

پاسخ سوال ۹:

از آنجایی که در صورت سوال قیدی برای نوع اشکال ذکر نشده است، تمام اشکالات را بررسی خواهیم کرد.

۱. نام متغیر negative مناسب نیست؛ زیرا با توجه به ادامه کد، این متغیر هیچ کجا به عنوان منفی عدد دیگری استفاده نشده است. این خطا اگرچه از کامپایل برنامه جلوگیری نمی‌کند اما از برنامه‌نویسی حرفه‌ای به دور است.

۲. متغیر x از نوع int است در حالیکه مقدار بازگشتی cin یک عدد int نیست. این خطا از کامپایل برنامه جلوگیری می‌کند لذا خطای زمان کامپایل است. از طرفی خطا به خاطر قرارگیری متغیر یک نوع در متغیر نوع دیگر است پس خطای نوع داده (زیر مجموعه خطای زمان کامپایل محسوب می‌شود.)

۳. در خط ۷ متغیر y قبل از مقداردهی استفاده شده است. به دلیل عدم مقداردهی قبلی y نتایجی که وابسته به y باشند ممکن است غلط شوند. این خطا یک خطای منطقی است که زیر مجموعه خطای زمان اجرا است.

٤. در خط ۸ از X (ایکس بزرگ) استفاده شده است در حالیکه متغیری به این نام وجود ندارد. دقت شود که C و مشتقات آن به کوچکی و بزرگی حروف حساس هستند. این خطا، خطای نحوی از زیر مجموعه‌های زمان کامپایل است.

٥. جهت عملگر در cout خط ۹ نادرست است. عملگر باید به صورت >> باشد. این خطای نحوی است.