**سوال 1 میان ترم) STACK\_BASED\_ALU**

**ماژول‌ها:**

1. STACK\_BASED\_ALU
2. tb4\_STACK\_BASED\_ALU
3. tb8\_STACK\_BASED\_ALU
4. tb16\_STACK\_BASED\_ALU
5. tb32\_STACK\_BASED\_ALU
6. InfixToPostfix
7. PostfixEvaluator
8. ExpressionEvaluator (testbench برای ماژول 6 و 7)

**ماژول 6 تا 8 برای قسمت ب سوال می‌باشد.**

**در ادامه داک، کد وریلاگ ماژول ها به ترتیب شماره نوشته شده، شرح داده می‌شوند.**

**ماژول STACK\_BASED\_ALU** **با توضیحات:**

Width of the data

Multiply

**Check for overflow**

Addition

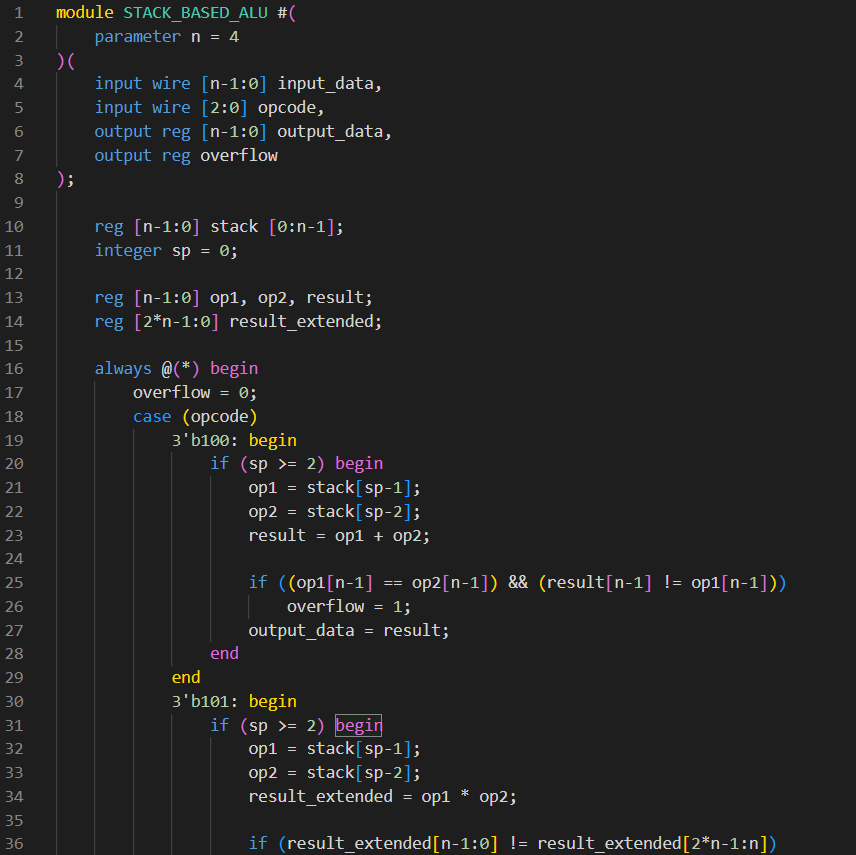
Temporary variables for operations

For detecting overflow in multiplication

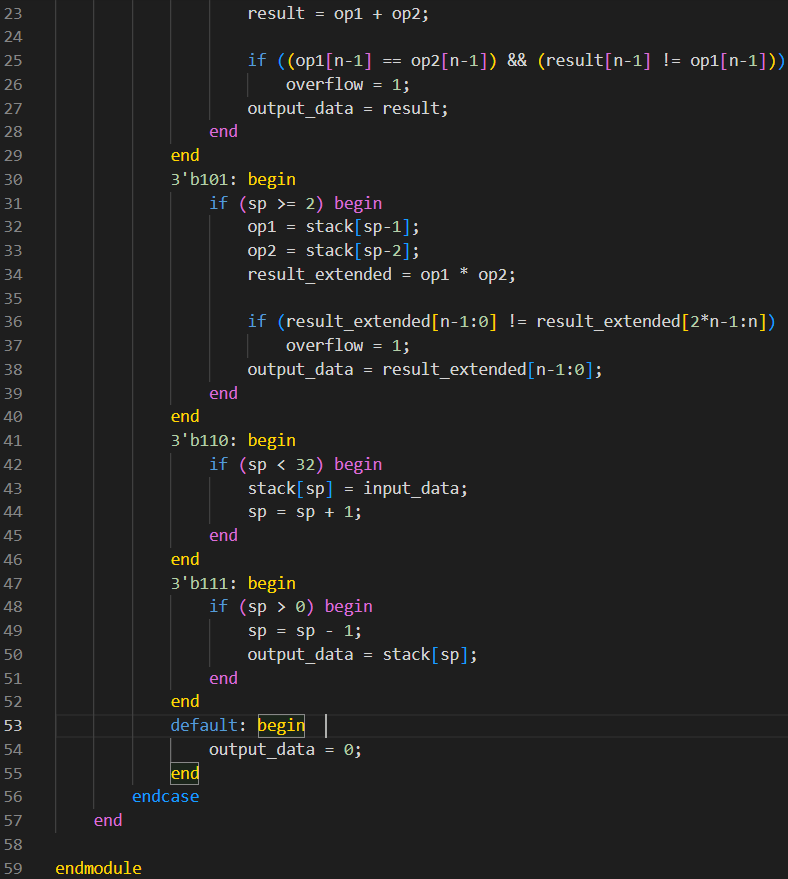
Stack for storing data

Stack pointer

Let's assume a stack size of n



**ادامه ماژول STACK\_BASED\_ALU :**



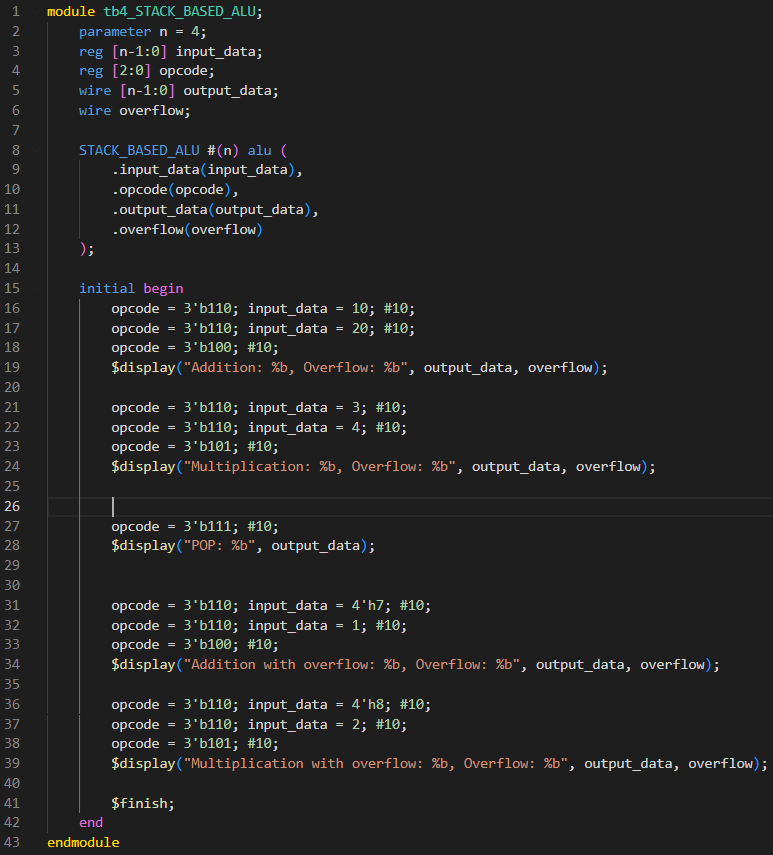
No Operation

PUSH

POP

**Check for overflow**

**توضیحات ماژول testbench برای ماژول :** STACK\_BASED\_ALU

****

Test overflow detection in multiplication

PUSH Max negative int

PUSH 2

PUSH 1

PUSH Max positive int

PUSH 4

PUSH 3

PUSH 20

PUSH 10

MULTIPLY

Test overflow detection in addition

ADD

**Test POP**

**Test multiplication**

ADD

MULTIPLY

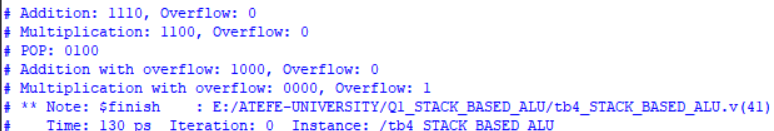
**Test addition**

**Changing this parameter to test different n(4, 8, 16, 32)**

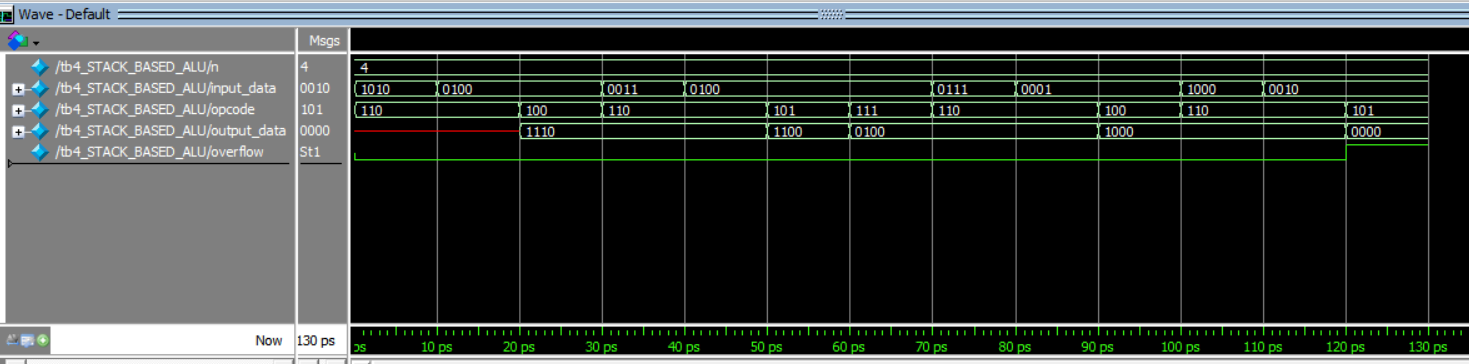
Instantiate the ALU module

**نمایش خروجی در transcript و waveform:**

* **tb4\_STACK\_BASED\_ALU ) ( parameter n = 4**



1. طبق ماژول تست بنچ که در صفحه پیش جز به جز توضیح داده شد، در ابتدا مقدار opcode، 110 است که دو مقدار 10و20 در استک پوش میشوند، سپس مقدار opcode، 100 میشود که یعنی جمع کردن دو مقدار بالا پشته، پس همانطور که میبینیم در تست کیس اول حاصل30 شده است. )این نکته که در سوال آمده رعایت شده "عملوندهاي Opcode ضرب و جمع دو عدد بالایی پشته است" (
2. سپس مجدد opcode، 110 میشود و 3 و 4 در پشته پوش میشوند، حال opcode، 101 میشود که باید دو مقدار بالا پشته را در هم ضرب کند که حاصل در خط دوم transcript، 1100 = 12 شده است.
3. حال opcode، 111 میشود که یعنی پاپ کردن از پشته، با توجه به اینکه در مرحله قبل ابتدا 3 پوش شد و سپس 4 این به این معنا است که بالاترین داده در استک 4 هست، همانطور که مشاهده می‌شود مقداری نیز که پاپ شده 100 = 4 است. ) پس این نکته سوال که " نتیجه عملیات ضرب/جمع در خروجی ماژول در دسترس خواهند بود و تغییري در پشته ایجاد نخواهند کرد." رعایت شده است. چون در مرحله قبل 3 را در 4 ضرب کردیم و حالا که اپکود 111 شد همان مقدار 4 که در پشته بود پاپ شد 🡨 تغییری در پشته ایجاد نشد.(
4. خط 4و5 در transcript نیز چک کردن اورفلو هست که همانطور که پیداست هنگام ضرب چون بیشترین اینتیجر منفی در 2 ضرب میشود اورفلو داریم که مقدار overflow ، 1 شده است.



Overflow =1

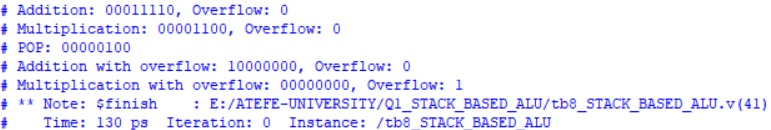
پاپ که خروجی 4 (**0100**)

آپکود **101** می شود پس خروجی حاصل ضرب دو عدد پوش شده قبلی که 3 و 4 بود یعنی 12 (**1100**)

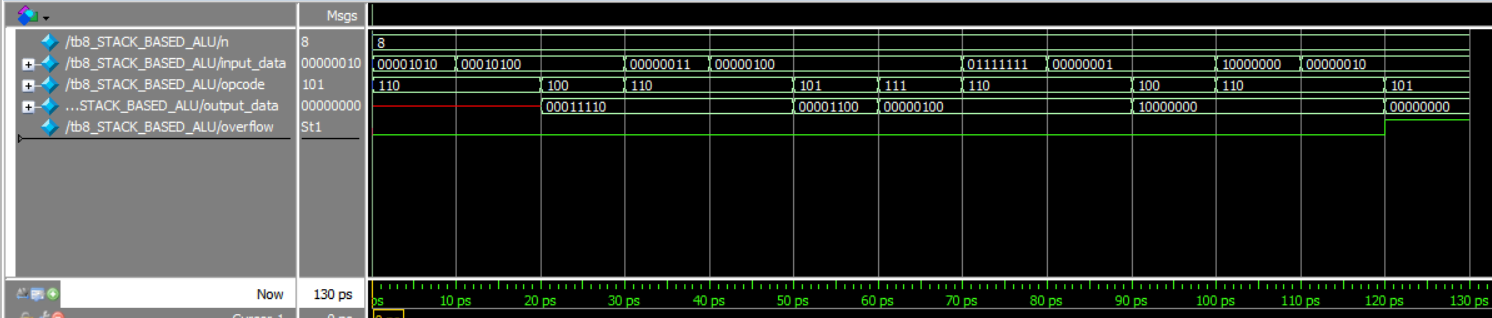
اپکود **100** می شود پس جمع 10و 20 که 30 (**1110**) شده است.

آپکود **110** هست دو عدد پوش می شوند.

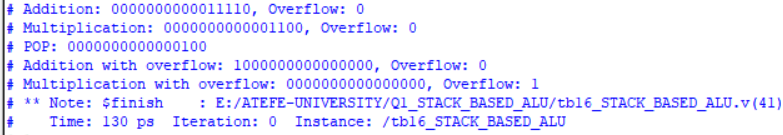
* **tb8\_STACK\_BASED\_ALU ) ( parameter n = 8**

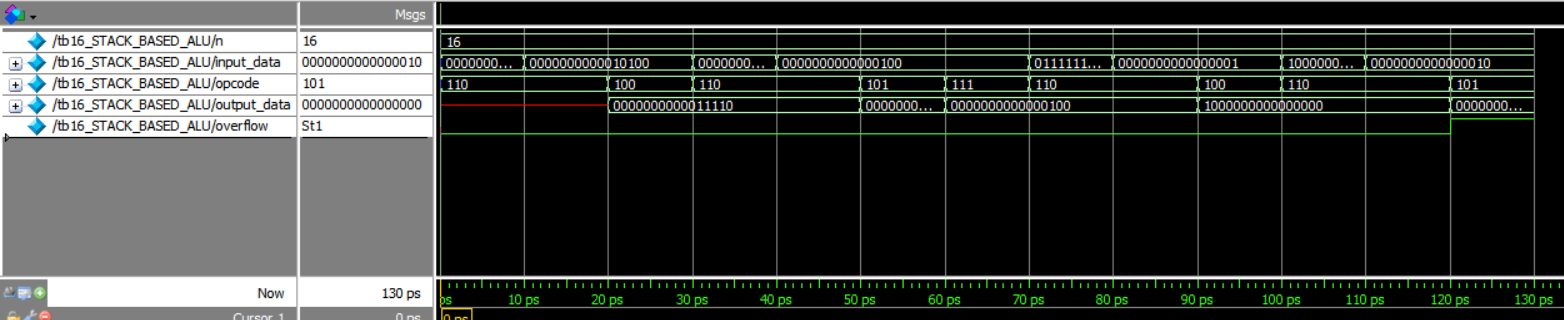


توضیحات و عملکرد عینا همانند خروجی های قبلی است (هنگامی که پارامتر 4 است) تنها خروجی ها 8 بیتی هستند.

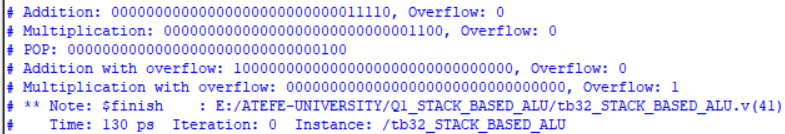


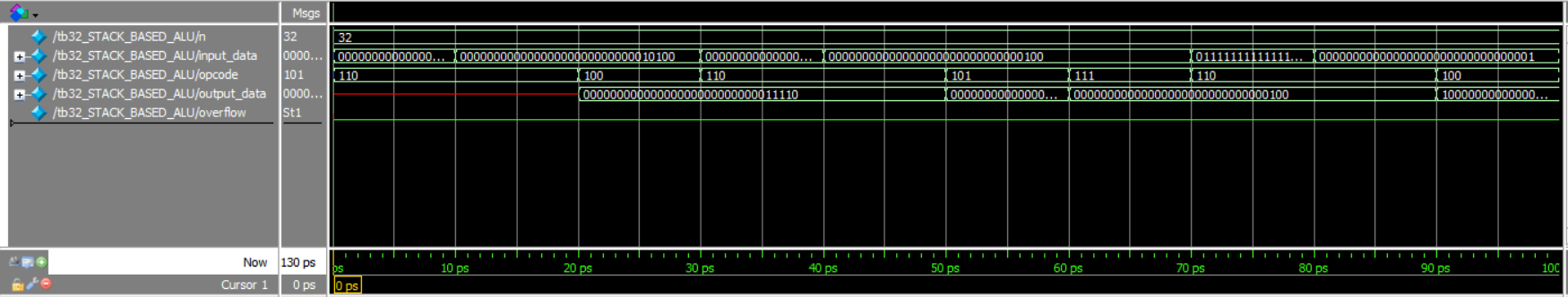
* **tb16\_STACK\_BASED\_ALU ) ( parameter n = 16**





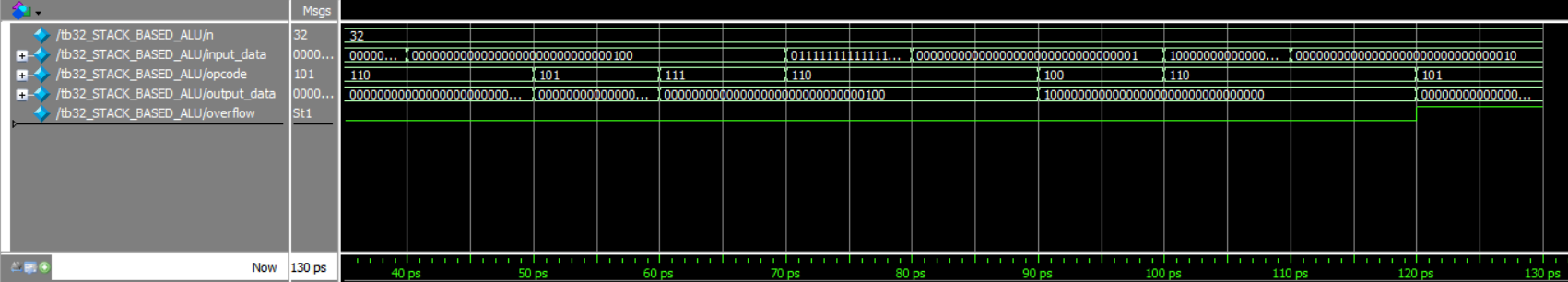
* **tb32\_STACK\_BASED\_ALU ) ( parameter n = 32**





**Waveform تا 100 ps**

**Waveform از 100ps به بعد :**



**قسمت ب)**

برای نوشتن یک ماژول که بتواند عبارات را به پسوندی تبدیل کرده و سپس محاسبه کند، ابتدا باید دو مرحله را انجام دهیم:

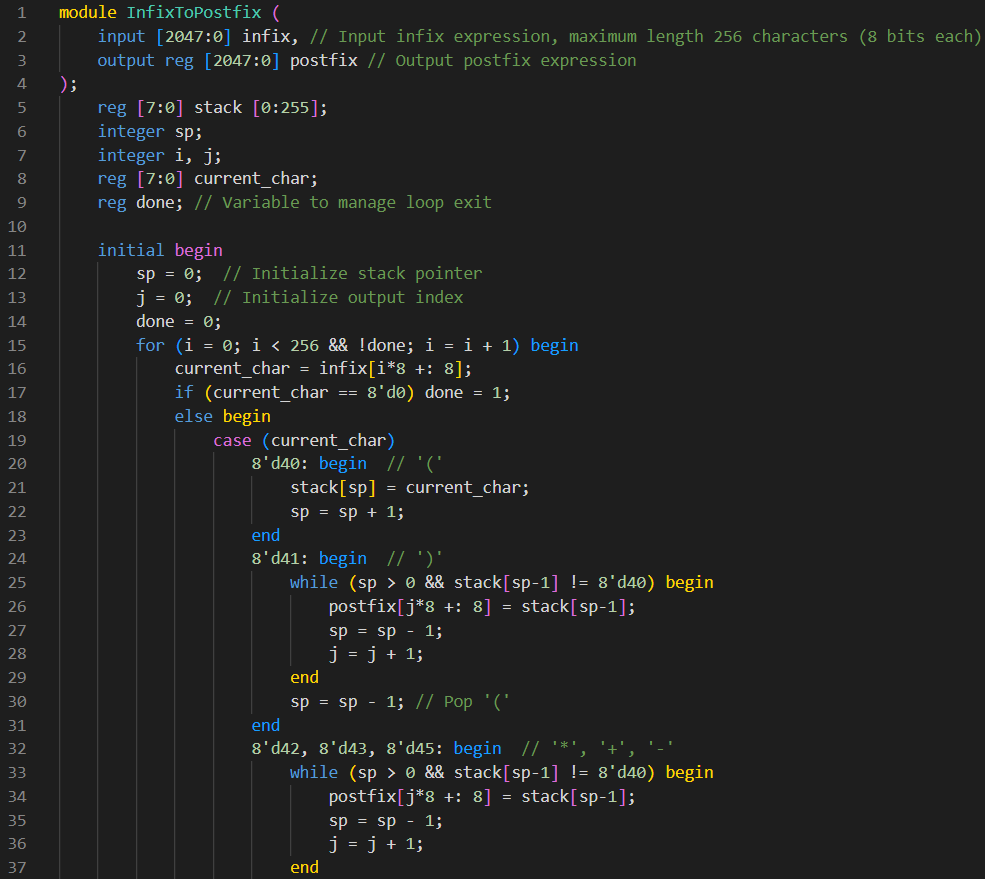
1. تبدیل عبارت میانوندی (infix) به عبارت پسوندی (postfix).
2. استفاده از ماژول STACK\_BASED\_ALU برای محاسبه نتیجه عبارت پسوندی.

ابتدا یک ماژول برای تبدیل عبارت میانوندی به پسوندی نوشته می‌شود. سپس با استفاده از یک ماژول دیگر، محاسبه نتیجه عبارت پسوندی انجام می‌گردد.

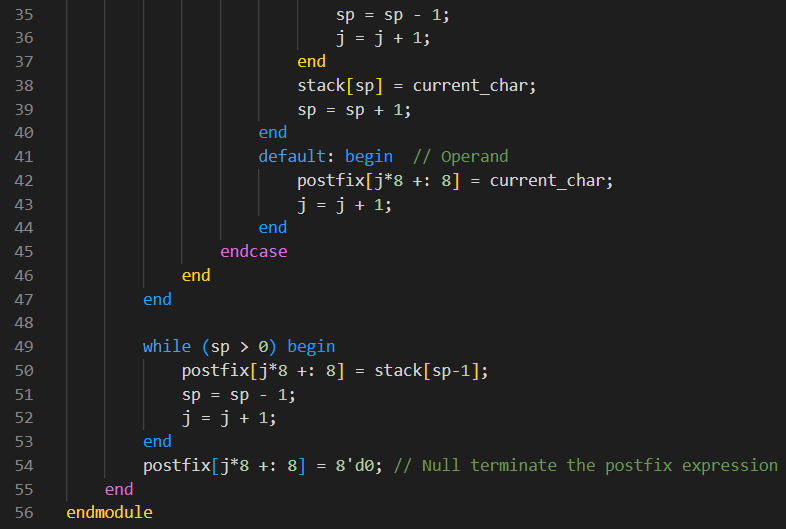
**ماژول تبدیل عبارت میانوندی به پسوندی InfixToPostfix**

در این قسمت از یک الگوریتم مانند الگوریتم شانتینگ-یارد (Shunting Yard) برای تبدیل عبارت میانوندی به پسوندی استفاده می‌کنیم.

جزئیات بیشتر ماژول به صورت کامنت گذاری در کد:



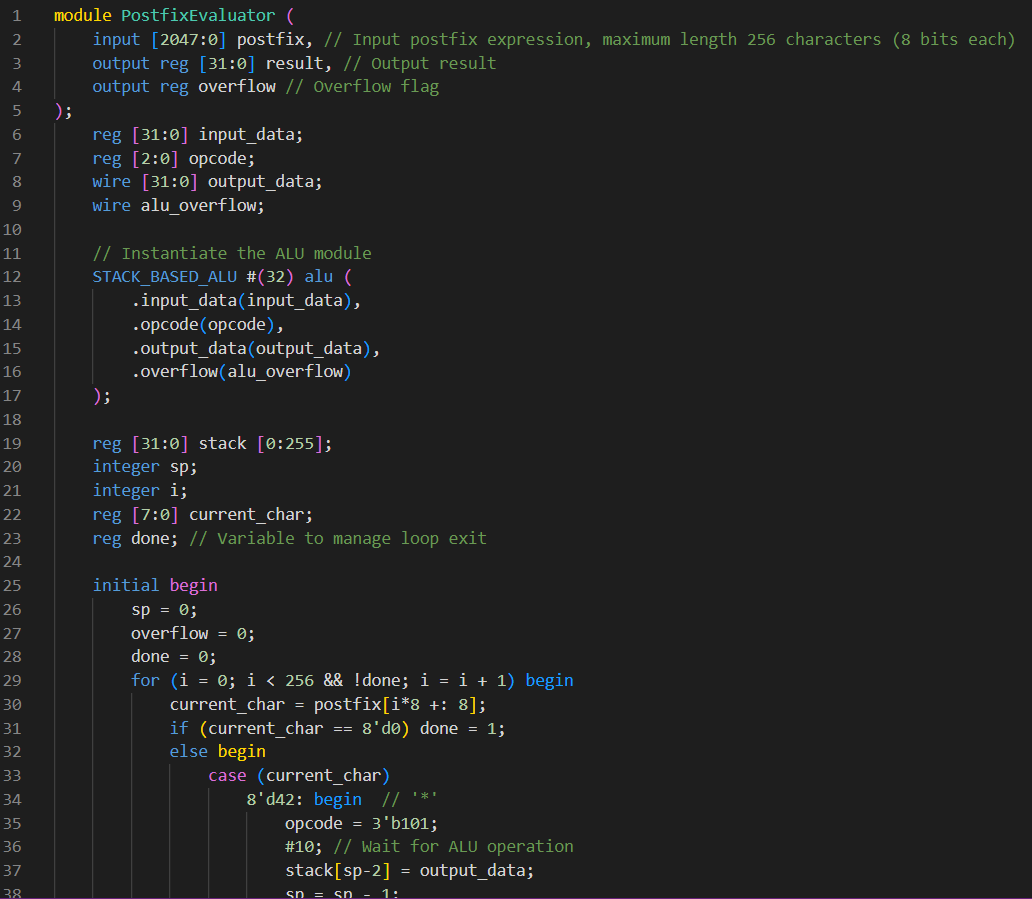
**ادامه ماژول InfixToPostfix**

****

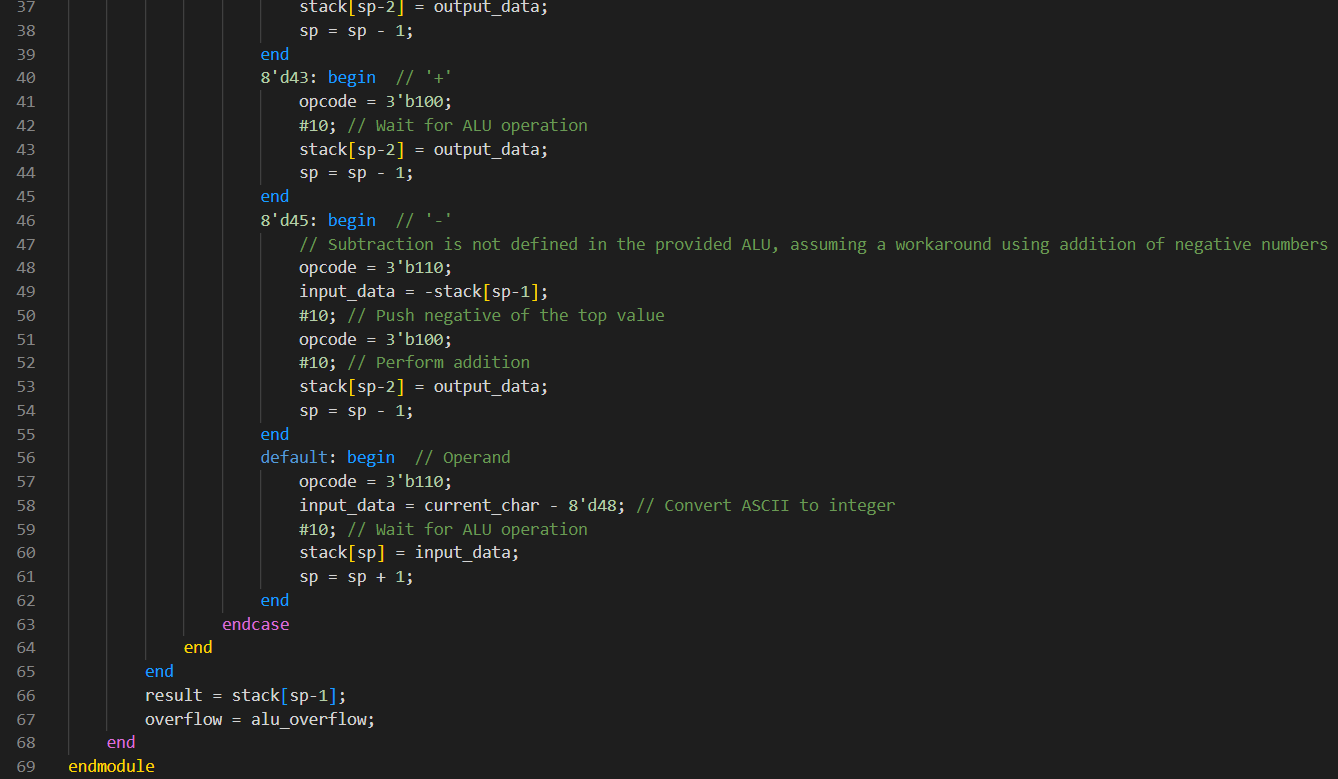
**ماژول محاسبه عبارت پسوندی PostfixEvaluator**

در این قسمت از ماژول STACK\_BASED\_ALU برای محاسبه نتیجه عبارت پسوندی استفاده می‌شود.

جزئیات بیشتر ماژول به صورت کامنت گذاری در کد:



ادامه ماژول **PostfixEvaluator:**



**ماژول ترکیبی ExpressionEvaluator**

ماژول نهایی که از دو ماژول قبلی استفاده می‌کند.

جزئیات بیشتر ماژول به صورت کامنت گذاری در کد:

