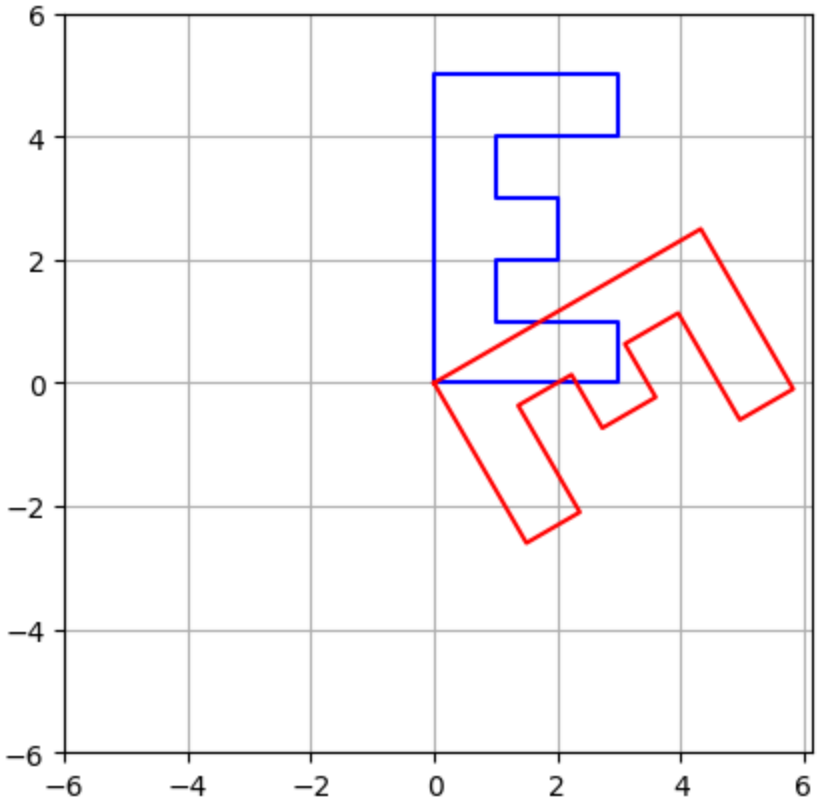
In [1]:

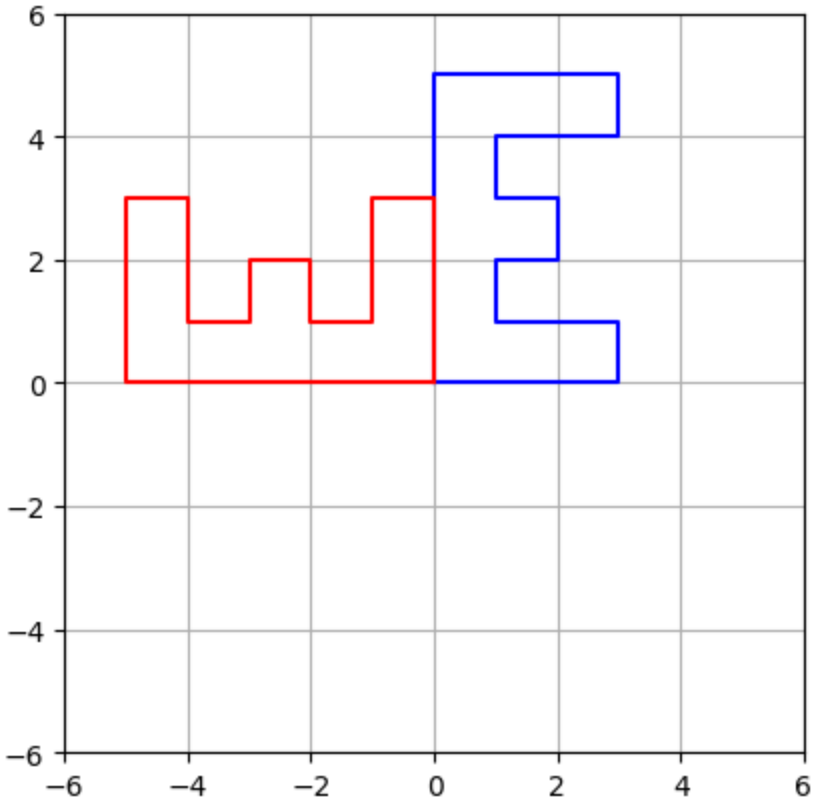
```
import numpy as np
import matplotlib .pyplot as plt

E = np.array ([(0 ,0 ,1) ,(3,0,1) ,(3,1,1) ,(1,1,1) ,(1,2,1) ,(2,2,1),
(2 ,3 ,1) ,(1,3,1) ,(1,4,1) ,(3,4,1) ,(3,5,1) ,(0,5,1),(0,0,1)])

# Matriz Identidad
Ic = np.eye (3)

#Matriz de Reflexión con respecto al eje x
Refx=np.array ([[1. , 0, 0],[0, -1., 0] ,[0. , 0., 1.]])
#Matriz de Reflexión con respecto al eje y
Refy=np.array([[-1., 0, 0],[0, 1., 0] ,[0. , 0., 1.]])
# Matriz de Rotación
theta= np.pi/3 #Ángulo de rotación deseado ,en este caso usamos pi/3.
R=np.array ([[ np.cos(theta), np.sin(theta), 0.],
[-np.sin(theta), np.cos(theta), 0.],
[0., 0., 1.]])
# Matriz de cambio de escala
s=2 #Escalar
S=np.array ([[s, 0, 0.], [0., s, 0.] ,[0. , 0., 1.]])
# Matriz de deformación horizontal/vertical
h=-1
v= 2
D=np.array ([[1. , h, 0.] ,[v, 1., 0.] ,[0. , 0., 1.]])
# Matriz de traslación
tx = -3
ty = -5
T=np.array ([[1. , 0, tx], [0, 1., ty], [0., 0., 1.]])
#transformación y plot
ax = plt.gca () #gca= get current axes , obtiene los ejes actuales
Ex = [] #Lista de primeras componentes
Ey = [] #Lista de segundas componentes
for row in E:
    output_rown = (R.dot(row)) #Multiplicación de la matriz por el punto respectivo

    x, y, _= output_rown #Coordenadas ya transformadas
    Ex.append(x) #Se guardan las coordenadas
    Ey.append(y)
plt.plot(E[:,0], E[:,1], color="blue") #Se grafica E sin transformar
plt.plot(Ex , Ey , color="Red") #Se grafica E transformada
ax. set_xticks (np.arange(-6, 8, 2)) # Fija el rango del eje x
ax. set_yticks (np.arange(-6, 8, 2)) # Fija el rango del eje y
plt.gca (). set_aspect ('equal', adjustable ='box') #Establece misma escala
plt.grid () #Habilita la cuadrícula
plt.show()
```



In [2]:

```
import numpy as np
import matplotlib .pyplot as plt

E = np.array ([(0 ,0 ,1) ,(3,0,1) ,(3,1,1) ,(1,1,1) ,(1,2,1) ,(2,2,1),
(2 ,3 ,1) ,(1,3,1) ,(1,4,1) ,(3,4,1) ,(3,5,1) ,(0,5,1),(0,0,1)])

# Matriz Identidad
Ic = np.eye (3)

#Matriz de Reflexión con respecto al eje x
Refx=np.array ([[1. , 0, 0],[0, -1., 0] ,[0. , 0., 1.]])
#Matriz de Reflexión con respecto al eje y
Refy=np.array([[-1., 0, 0],[0, 1., 0] ,[0. , 0., 1.]])
# Matriz de Rotación
theta= np.pi*3/2 #Ángulo de rotación deseado ,en este caso usamos pi/3.
R=np.array ([[ np.cos(theta), np.sin(theta), 0.],
[-np.sin(theta), np.cos(theta), 0.],
[0., 0., 1.]])
# Matriz de cambio de escala
s=2 #Escalar
S=np.array ([[s, 0, 0.], [0., s, 0.] ,[0. , 0., 1.]])
# Matriz de deformación horizontal/vertical
h=-1
v= 2
D=np.array ([[1. , h, 0.] ,[v, 1., 0.] ,[0. , 0., 1.]])
# Matriz de traslación
tx = -3
ty = -5
T=np.array ([[1. , 0, tx], [0, 1., ty], [0., 0., 1.]])
#transformación y plot
ax = plt.gca () #gca= get current axes , obtiene los ejes actuales
Ex = [] #Lista de primeras componentes
Ey = [] #Lista de segundas componentes
for row in E:
    output_rown = (R.dot(row)) #Multiplicación de la matriz por el punto respectivo
```

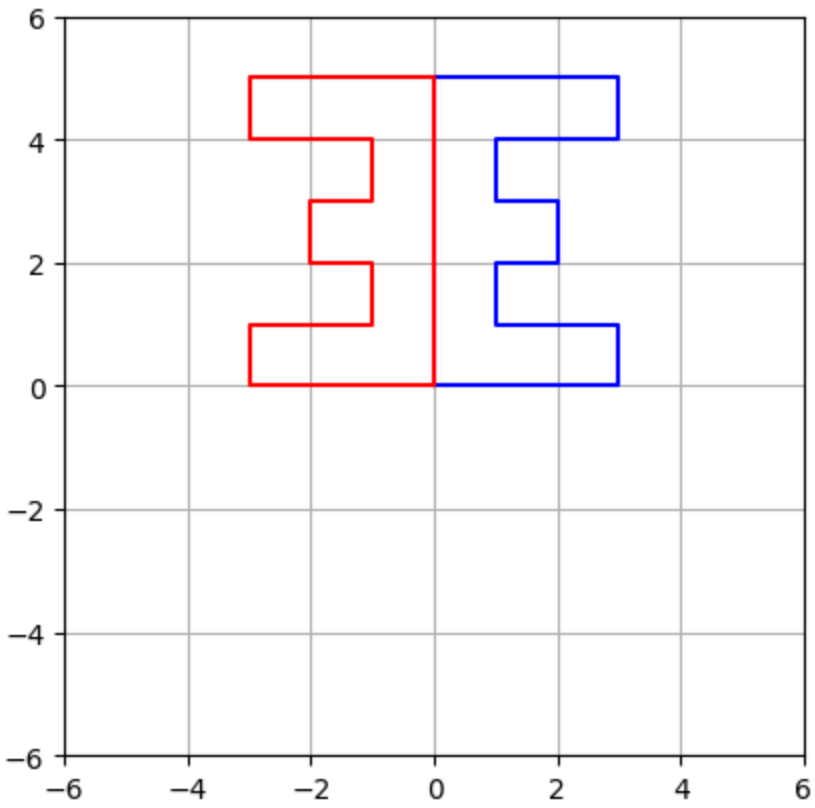
```
    x, y, _= output_rown #Coordenadas ya transformadas
    Ex.append(x) #Se guardan las coordenadas
    Ey.append(y)
plt.plot(E[:,0], E[:,1], color="blue") #Se grafica E sin transformar
plt.plot(Ex , Ey , color="Red") #Se grafica E transformada
ax. set_xticks (np.arange(-6, 8, 2)) # Fija el rango del eje x
ax. set_yticks (np.arange(-6, 8, 2)) # Fija el rango del eje y
plt.gca (). set_aspect ('equal', adjustable ='box') #Establece misma escala
plt.grid () #Habilita la cuadrícula
plt.show()
```



In [6]:

```
import numpy as np
import matplotlib .pyplot as plt

E = np.array ([(0 ,0 ,1) ,(3,0,1) ,(3,1,1) ,(1,1,1) ,(1,2,1) ,(2,2,1),
(2 ,3 ,1) ,(1,3,1) ,(1,4,1) ,(3,4,1) ,(3,5,1) ,(0,5,1),(0,0,1)])

# Matriz Identidad
Ic = np.eye (3)
#Matriz de Reflexión con respecto al eje x
Refx=np.array ([[1. , 0, 0],[0, -1., 0] ,[0. , 0., 1.]])
#Matriz de Reflexión con respecto al eje y
Refy=np.array([[-1., 0, 0],[0, 1., 0] ,[0. , 0., 1.]])
# Matriz de Rotación
theta= np.pi*3/2 #Ángulo de rotación deseado ,en este caso usamos pi/3.
R=np.array ([[ np.cos(theta), np.sin(theta), 0.],
[-np.sin(theta), np.cos(theta), 0.],
[0., 0., 1.]])
# Matriz de cambio de escala
s=2 #Escalar
S=np.array ([[s, 0, 0.], [0., s, 0.] ,[0. , 0., 1.]])
# Matriz de deformación horizontal/vertical
h=-1
v= 2
D=np.array ([[1. , h, 0.] ,[v, 1., 0.] ,[0. , 0., 1.]])
# Matriz de traslación
```

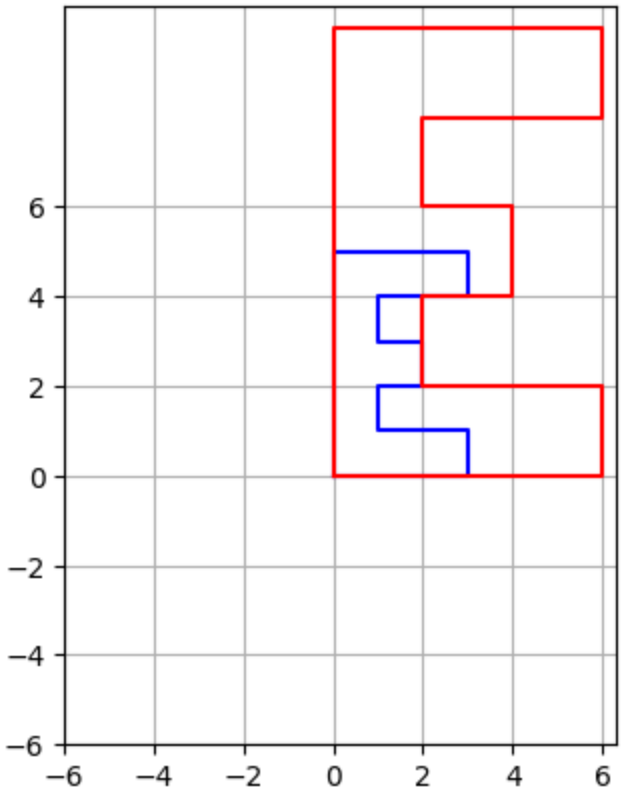
```
tx = -3
ty = -5
T=np.array ([[1. , 0, tx], [0, 1., ty], [0., 0., 1.]])
#transformación y plot
ax = plt.gca () #gca= get current axes , obtiene los ejes actuales
Ex = [] #Lista de primeras componentes
Ey = [] #Lista de segundas componentes
for row in E:
    output_rown = (Refy.dot(row)) #Multiplicación de la matriz por el punto respectivo

    x, y, _= output_rown #Coordenadas ya transformadas
    Ex.append(x) #Se guardan las coordenadas
    Ey.append(y)
plt.plot(E[:,0], E[:,1], color="blue") #Se grafica E sin transformar
plt.plot(Ex , Ey , color="Red") #Se grafica E transformada
ax. set_xticks (np.arange(-6, 8, 2)) # Fija el rango del eje x
ax. set_yticks (np.arange(-6, 8, 2)) # Fija el rango del eje y
plt.gca (). set_aspect ('equal', adjustable ='box') #Establece misma escala
plt.grid () #Habilita la cuadrícula
plt.show()
```



In [5]:

```
import numpy as np
import matplotlib .pyplot as plt

E = np.array ([(0 ,0 ,1) ,(3,0,1) ,(3,1,1) ,(1,1,1) ,(1,2,1) ,(2,2,1),
(2 ,3 ,1) ,(1,3,1) ,(1,4,1) ,(3,4,1) ,(3,5,1) ,(0,5,1),(0,0,1)])

# Matriz Identidad
Ic = np.eye (3)
#Matriz de Reflexión con respecto al eje x
Refx=np.array ([[1. , 0, 0],[0, -1., 0] ,[0. , 0., 1.]])
#Matriz de Reflexión con respecto al eje y
Refy=np.array([[-1., 0, 0],[0, 1., 0] ,[0. , 0., 1.]])
# Matriz de Rotación
theta= np.pi*3/2 #Ángulo de rotación deseado ,en este caso usamos pi/3.
R=np.array ([[ np.cos(theta), np.sin(theta), 0.],
```

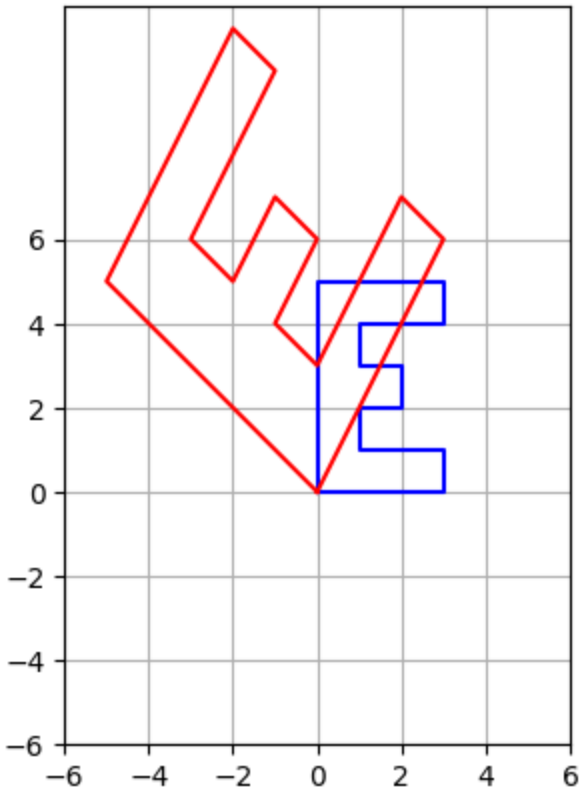
```
[-np.sin(theta), np.cos(theta), 0.],
[0., 0., 1.]])
# Matriz de cambio de escala
s=2 #Escalar
S=np.array ([[s, 0, 0.], [0., s, 0.] ,[0. , 0., 1.]])
# Matriz de deformación horizontal/vertical
h=-1
v= 2
D=np.array ([[1. , h, 0.] ,[v, 1., 0.] ,[0. , 0., 1.]])
# Matriz de traslación
tx = -3
ty = -5
T=np.array ([[1. , 0, tx], [0, 1., ty], [0., 0., 1.]])
#transformación y plot
ax = plt.gca () #gca= get current axes , obtiene los ejes actuales
Ex = [] #Lista de primeras componentes
Ey = [] #Lista de segundas componentes
for row in E:
    output_rown = (S.dot(row)) #Multiplicación de la matriz por el punto respectivo

    x, y, _= output_rown #Coordenadas ya transformadas
    Ex.append(x) #Se guardan las coordenadas
    Ey.append(y)
plt.plot(E[:,0], E[:,1], color="blue") #Se grafica E sin transformar
plt.plot(Ex , Ey , color="Red") #Se grafica E transformada
ax. set_xticks (np.arange(-6, 8, 2)) # Fija el rango del eje x
ax. set_yticks (np.arange(-6, 8, 2)) # Fija el rango del eje y
plt.gca (). set_aspect ('equal', adjustable ='box') #Establece misma escala
plt.grid () #Habilita la cuadrícula
plt.show()
```



In [4]:

```
import numpy as np
import matplotlib .pyplot as plt

E = np.array ([(0 ,0 ,1) ,(3,0,1) ,(3,1,1) ,(1,1,1) ,(1,2,1) ,(2,2,1),
(2 ,3 ,1) ,(1,3,1) ,(1,4,1) ,(3,4,1) ,(3,5,1) ,(0,5,1),(0,0,1)])
```

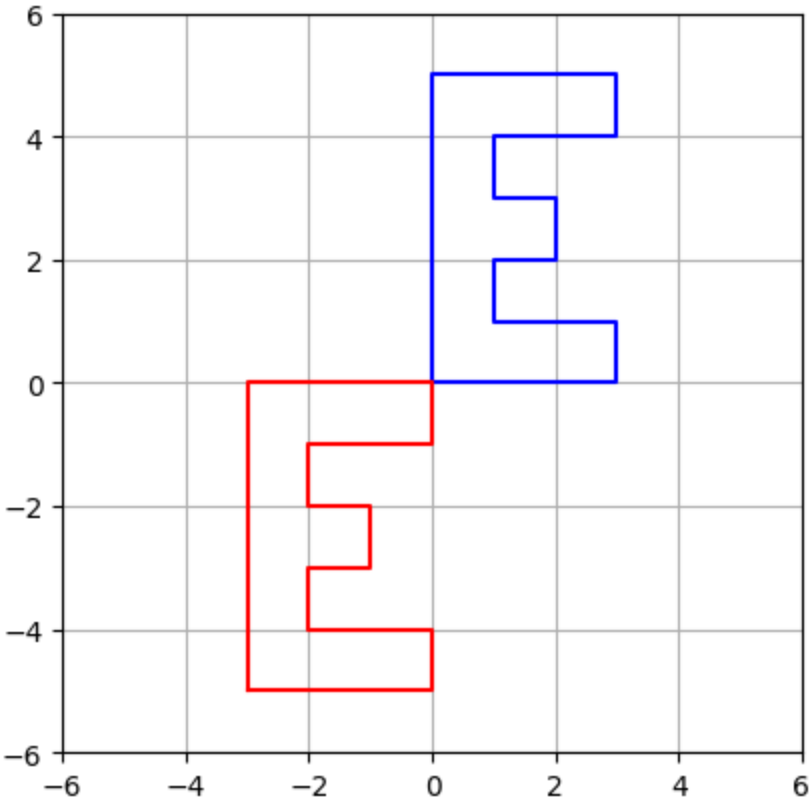
```
# Matriz Identidad
Ic = np.eye (3)
#Matriz de Reflexión con respecto al eje x
Refx=np.array ([[1. , 0, 0],[0, -1., 0] ,[0. , 0., 1.]])
#Matriz de Reflexión con respecto al eje y
Refy=np.array([[-1., 0, 0],[0, 1., 0] ,[0. , 0., 1.]])
# Matriz de Rotación
theta= np.pi*3/2 #Ángulo de rotación deseado ,en este caso usamos pi/3.
R=np.array ([[ np.cos(theta), np.sin(theta), 0.],
[-np.sin(theta), np.cos(theta), 0.],
[0., 0., 1.]])
# Matriz de cambio de escala
s=2 #Escalar
S=np.array ([[s, 0, 0.], [0., s, 0.] ,[0. , 0., 1.]])
# Matriz de deformación horizontal/vertical
h=-1
v= 2
D=np.array ([[1. , h, 0.] ,[v, 1., 0.] ,[0. , 0., 1.]])
# Matriz de traslación
tx = -3
ty = -5
T=np.array ([[1. , 0, tx], [0, 1., ty], [0., 0., 1.]])
#transformación y plot
ax = plt.gca () #gca= get current axes , obtiene los ejes actuales
Ex = [] #Lista de primeras componentes
Ey = [] #Lista de segundas componentes
for row in E:
    output_rown = (D.dot(row)) #Multiplicación de la matriz por el punto respectivo

    x, y, _= output_rown #Coordenadas ya transformadas
    Ex.append(x) #Se guardan las coordenadas
    Ey.append(y)
plt.plot(E[:,0], E[:,1], color="blue") #Se grafica E sin transformar
plt.plot(Ex , Ey , color="Red") #Se grafica E transformada
ax. set_xticks (np.arange(-6, 8, 2)) # Fija el rango del eje x
ax. set_yticks (np.arange(-6, 8, 2)) # Fija el rango del eje y
plt.gca (). set_aspect ('equal', adjustable ='box') #Establece misma escala
plt.grid () #Habilita la cuadrícula
plt.show()
```



In [3]:

```
import numpy as np
import matplotlib .pyplot as plt

E = np.array ([(0 ,0 ,1) ,(3,0,1) ,(3,1,1) ,(1,1,1) ,(1,2,1) ,(2,2,1),
(2 ,3 ,1) ,(1,3,1) ,(1,4,1) ,(3,4,1) ,(3,5,1) ,(0,5,1),(0,0,1)])

# Matriz Identidad
Ic = np.eye (3)
#Matriz de Reflexión con respecto al eje x
Refx=np.array ([[1. , 0, 0],[0, -1., 0] ,[0. , 0., 1.]])
#Matriz de Reflexión con respecto al eje y
Refy=np.array([[-1., 0, 0],[0, 1., 0] ,[0. , 0., 1.]])
# Matriz de Rotación
theta= np.pi*3/2 #Ángulo de rotación deseado ,en este caso usamos pi/3.
R=np.array ([[ np.cos(theta), np.sin(theta), 0.],
[-np.sin(theta), np.cos(theta), 0.],
[0., 0., 1.]])
# Matriz de cambio de escala
s=2 #Escalar
S=np.array ([[s, 0, 0.], [0., s, 0.] ,[0. , 0., 1.]])
# Matriz de deformación horizontal/vertical
h=-1
v= 2
D=np.array ([[1. , h, 0.] ,[v, 1., 0.] ,[0. , 0., 1.]])
# Matriz de traslación
tx = -3
ty = -5
T=np.array ([[1. , 0, tx], [0, 1., ty], [0., 0., 1.]])
#transformación y plot
ax = plt.gca () #gca= get current axes , obtiene los ejes actuales
Ex = [] #Lista de primeras componentes
Ey = [] #Lista de segundas componentes
for row in E:
    output_rown = (T.dot(row)) #Multiplicación de la matriz por el punto respectivo

    x, y, _= output_rown #Coordenadas ya transformadas
```

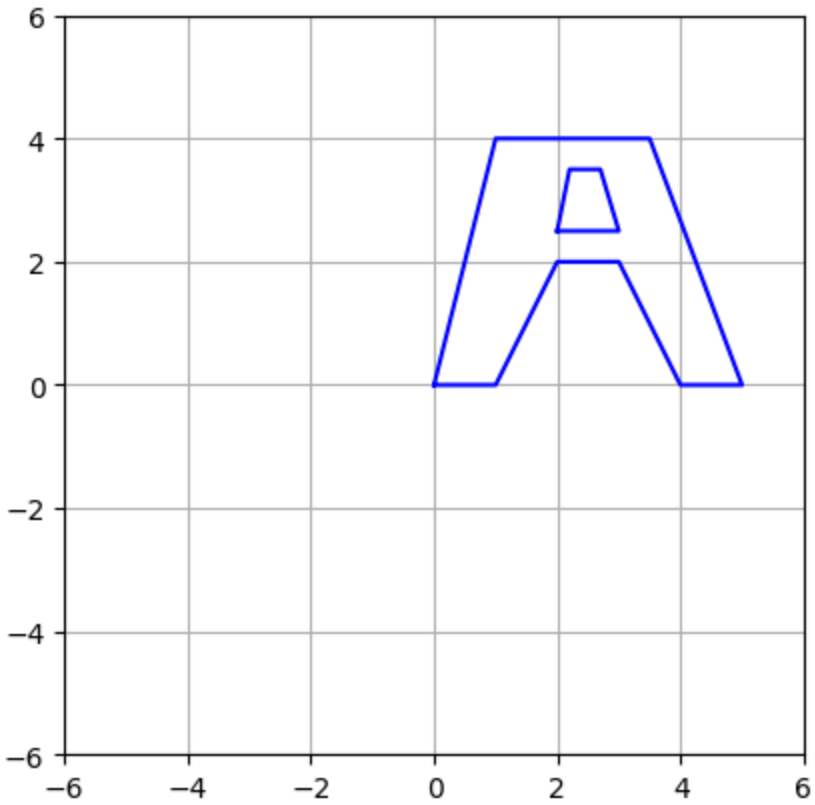
```
    Ex.append(x) #Se guardan las coordenadas
    Ey.append(y)
plt.plot(E[:,0], E[:,1], color="blue") #Se grafica E sin transformar
plt.plot(Ex , Ey , color="Red") #Se grafica E transformada
ax. set_xticks (np.arange(-6, 8, 2)) # Fija el rango del eje x
ax. set_yticks (np.arange(-6, 8, 2)) # Fija el rango del eje y
plt.gca (). set_aspect ('equal', adjustable ='box') #Establece misma escala
plt.grid () #Habilita la cuadrícula
plt.show()
```



In [48]:

```
import numpy as np
import matplotlib .pyplot as plt
A = np.array([
    (0,0,1),
    (1,0,1),
    (2,2,1),
    (3,2,1),
    (4,0,1),
    (5,0,1),
    (3.5,4,1),
    (1,4,1),
    (0,0,1)
])
A1 = np.array([
    (2,2.5,1),
    (3,2.5,1),
    (2.7,3.5,1),
    (2.2,3.5,1),
    (2,2.5,1)
])
Ax = []
Ay = []
A1x = []
A1y = []
plt.plot(A[:,0], A[:,1], color="blue")
plt.plot(A1[:,0], A1[:,1], color="blue")
```

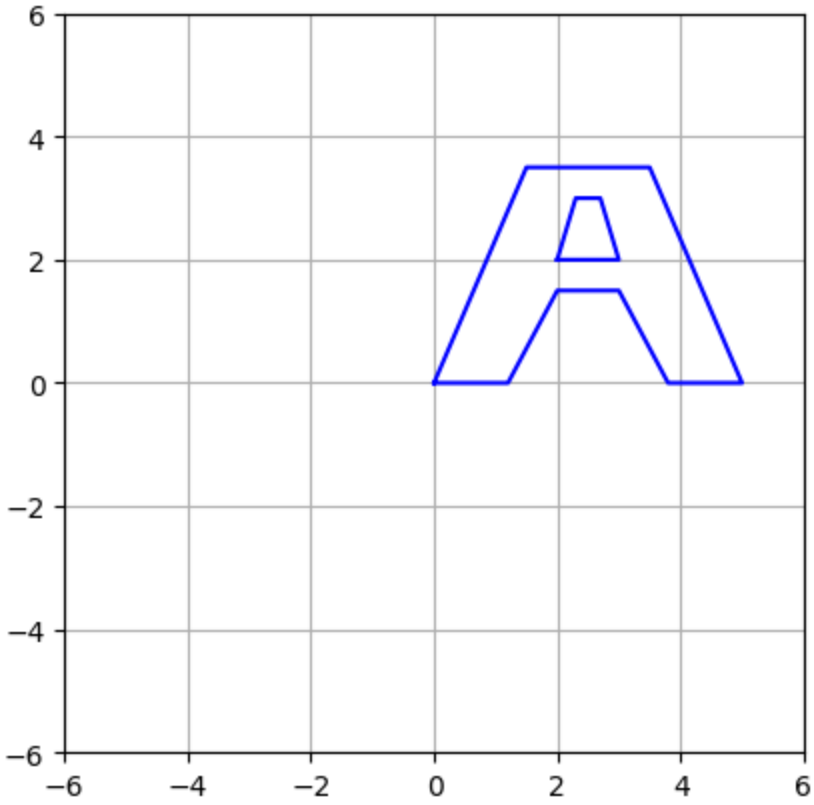
```
plt.plot(Ax, Ay, color="red")
plt.plot(A1x, A1y, color="red")
ax = plt.gca ()
ax. set_xticks (np.arange(-6, 8, 2))
ax. set_yticks (np.arange(-6, 8, 2))
plt.gca (). set_aspect ('equal', adjustable ='box')
plt.grid ()
plt.show()
```



In [63]:

```
import numpy as np
import matplotlib .pyplot as plt
A = np.array([
    (0,0,1),
    (1.2,0,1),
    (2,1.5,1),
    (3,1.5,1),
    (3.8,0,1),
    (5,0,1),
    (3.5,3.5,1),
    (1.5,3.5,1),
    (0,0,1)
])
A1 = np.array([
    (2,2,1),
    (3,2,1),
    (2.7,3,1),
    (2.3,3,1),
    (2,2,1)
])
Ax = []
Ay = []
A1x = []
A1y = []
plt.plot(A[:,0], A[:,1], color="blue")
plt.plot(A1[:,0], A1[:,1], color="blue")
plt.plot(Ax, Ay, color="red")
```

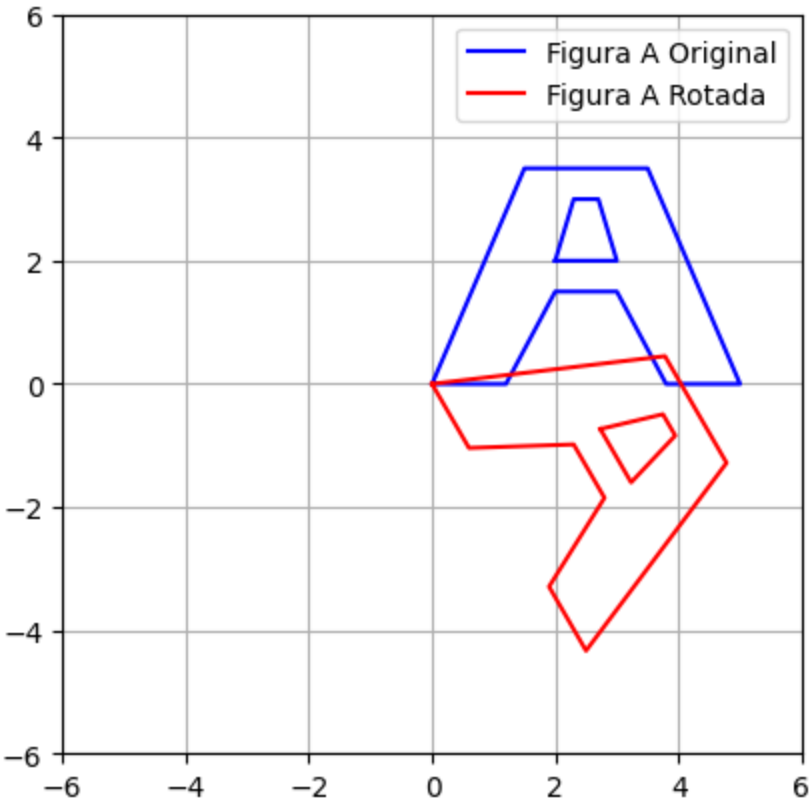
```
plt.plot(A1x, A1y, color="red")
ax = plt.gca ()
ax. set_xticks (np.arange(-6, 8, 2))
ax. set_yticks (np.arange(-6, 8, 2))
plt.gca (). set_aspect ('equal', adjustable ='box')
plt.grid ()
plt.show()
```



In [64]:

```
import numpy as np
import matplotlib.pyplot as plt

# Puntos de las figuras
A = np.array([
    (0,0,1),
    (1.2,0,1),
    (2,1.5,1),
    (3,1.5,1),
    (3.8,0,1),
    (5,0,1),
    (3.5,3.5,1),
    (1.5,3.5,1),
    (0,0,1)
])
A1 = np.array([
    (2,2,1),
    (3,2,1),
    (2.7,3,1),
    (2.3,3,1),
    (2,2,1)
])

# Matriz de rotación con theta = pi/3
theta = np.pi / 3
R = np.array([
    [np.cos(theta), np.sin(theta), 0],
    [-np.sin(theta), np.cos(theta), 0],
```

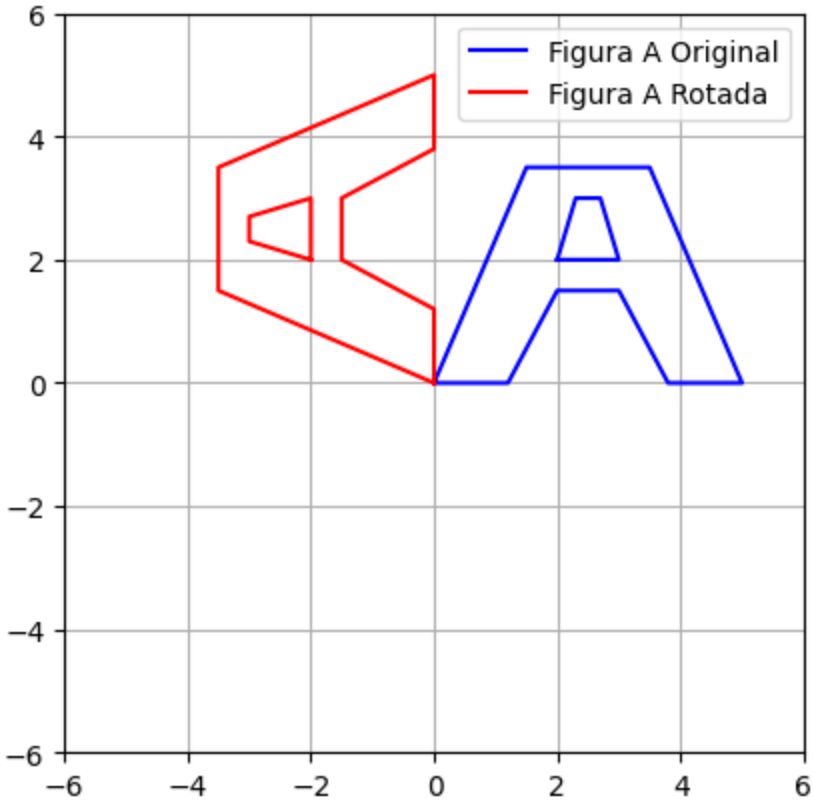
```
    [0, 0, 1]
])

# Aplicar la rotación a los puntos de A y A1
Ax, Ay = [], []
A1x, A1y = [], []

for row in A:
    output_row = R.dot(row)
    Ax.append(output_row[0])
    Ay.append(output_row[1])

for row in A1:
    output_row = R.dot(row)
    A1x.append(output_row[0])
    A1y.append(output_row[1])

# Graficar las figuras originales y transformadas
plt.plot(A[:, 0], A[:, 1], color="blue", label="Figura A Original")
plt.plot(A1[:, 0], A1[:, 1], color="blue")
plt.plot(Ax, Ay, color="red", label="Figura A Rotada")
plt.plot(A1x, A1y, color="red")

# Configuración del gráfico
ax = plt.gca()
ax.set_xticks(np.arange(-6, 8, 2))
ax.set_yticks(np.arange(-6, 8, 2))
plt.gca().set_aspect('equal', adjustable='box')
plt.grid()
plt.legend()
plt.show()
```



In [68]:

```
import numpy as np
import matplotlib.pyplot as plt

# Puntos de las figuras
```

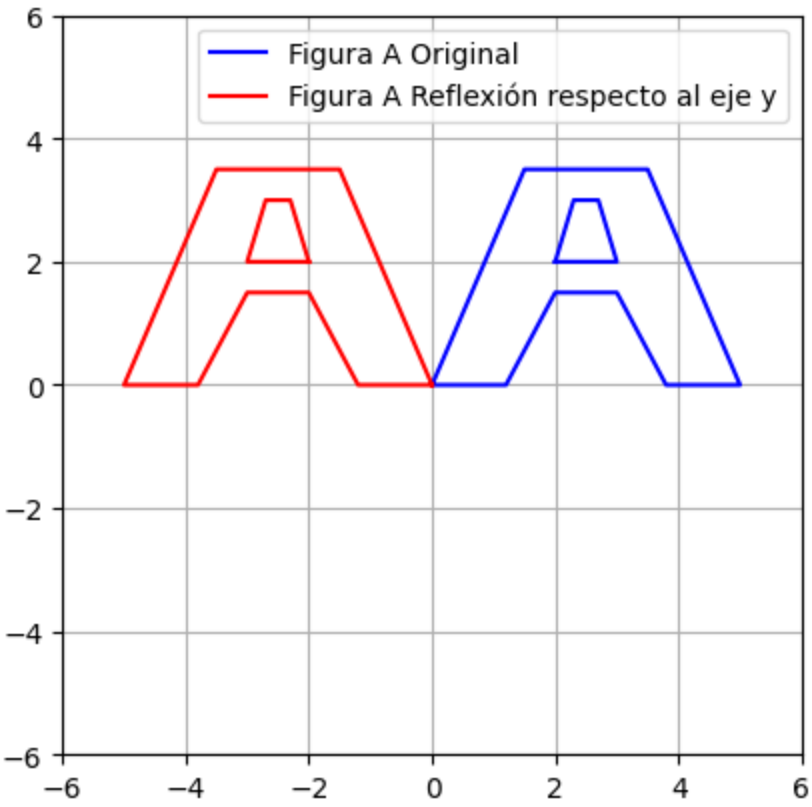
```
A = np.array([
    (0,0,1),
    (1.2,0,1),
    (2,1.5,1),
    (3,1.5,1),
    (3.8,0,1),
    (5,0,1),
    (3.5,3.5,1),
    (1.5,3.5,1),
    (0,0,1)
])
A1 = np.array([
    (2,2,1),
    (3,2,1),
    (2.7,3,1),
    (2.3,3,1),
    (2,2,1)
])

# Matriz de rotación con theta = pi/3
theta = 3 * np.pi/2
R = np.array([
    [np.cos(theta), np.sin(theta), 0],
    [-np.sin(theta), np.cos(theta), 0],
    [0, 0, 1]
])

# Aplicar la rotación a los puntos de A y A1
Ax, Ay = [], []
A1x, A1y = [], []

for row in A:
    output_row = R.dot(row)
    Ax.append(output_row[0])
    Ay.append(output_row[1])

for row in A1:
    output_row = R.dot(row)
    A1x.append(output_row[0])
    A1y.append(output_row[1])

# Graficar las figuras originales y transformadas
plt.plot(A[:, 0], A[:, 1], color="blue", label="Figura A Original")
plt.plot(A1[:, 0], A1[:, 1], color="blue")
plt.plot(Ax, Ay, color="red", label="Figura A Rotada")
plt.plot(A1x, A1y, color="red")

# Configuración del gráfico
ax = plt.gca()
ax.set_xticks(np.arange(-6, 8, 2))
ax.set_yticks(np.arange(-6, 8, 2))
plt.gca().set_aspect('equal', adjustable='box')
plt.grid()
plt.legend()
plt.show()
```



In [73]:

```
import numpy as np
import matplotlib.pyplot as plt

# Puntos de las figuras
A = np.array([
    (0,0,1),
    (1.2,0,1),
    (2,1.5,1),
    (3,1.5,1),
    (3.8,0,1),
    (5,0,1),
    (3.5,3.5,1),
    (1.5,3.5,1),
    (0,0,1)
])
A1 = np.array([
    (2,2,1),
    (3,2,1),
    (2.7,3,1),
    (2.3,3,1),
    (2,2,1)
])
Ref_y = np.array([
    [-1, 0, 0],
    [0, 1, 0],
    [0, 0, 1]
])

# Matriz de rotación con theta = pi/3
theta = 3 * np.pi/2
R = np.array([
    [np.cos(theta), np.sin(theta), 0],
    [-np.sin(theta), np.cos(theta), 0],
    [0, 0, 1]
])
```

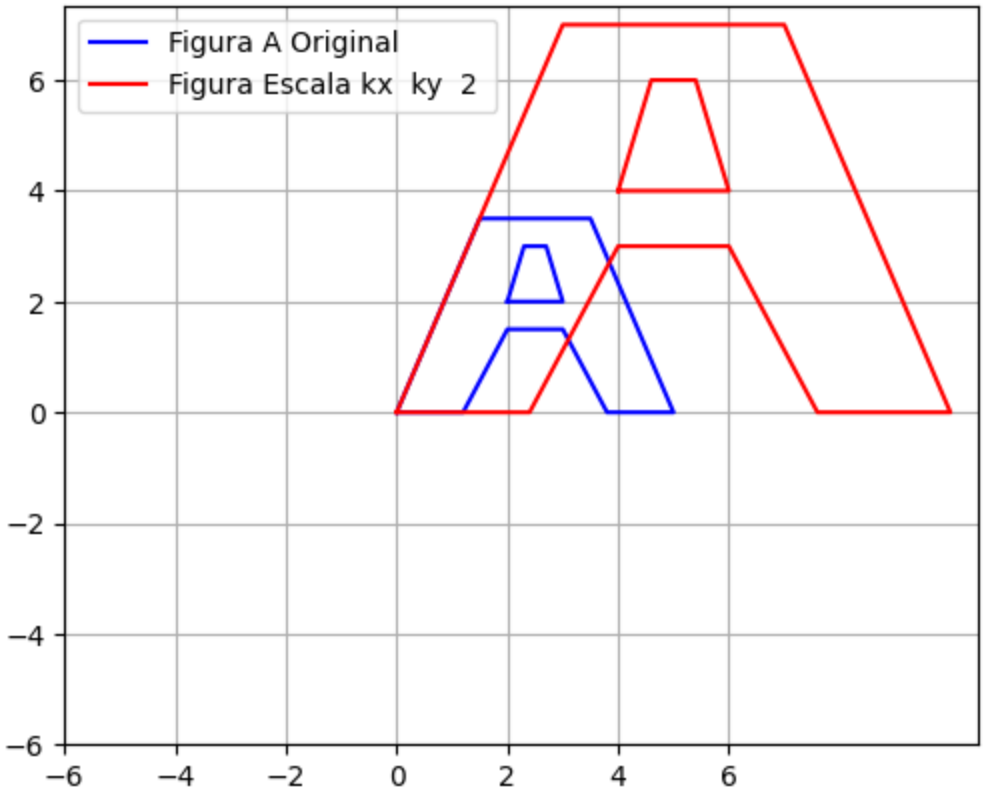
```
# Aplicar la rotación a los puntos de A y A1
Ax, Ay = [], []
A1x, A1y = [], []

for row in A:
    output_row = Ref_y.dot(row)
    Ax.append(output_row[0])
    Ay.append(output_row[1])

for row in A1:
    output_row = Ref_y.dot(row)
    A1x.append(output_row[0])
    A1y.append(output_row[1])

# Graficar las figuras originales y transformadas
plt.plot(A[:, 0], A[:, 1], color="blue", label="Figura A Original")
plt.plot(A1[:, 0], A1[:, 1], color="blue")
plt.plot(Ax, Ay, color="red", label="Figura A Reflexión respecto al eje y")
plt.plot(A1x, A1y, color="red")

# Configuración del gráfico
ax = plt.gca()
ax.set_xticks(np.arange(-6, 8, 2))
ax.set_yticks(np.arange(-6, 8, 2))
plt.gca().set_aspect('equal', adjustable='box')
plt.grid()
plt.legend()
plt.show()
```



In [77]:

```
import numpy as np
import matplotlib.pyplot as plt

# Puntos de las figuras
A = np.array([
    (0,0,1),
    (1.2,0,1),
```

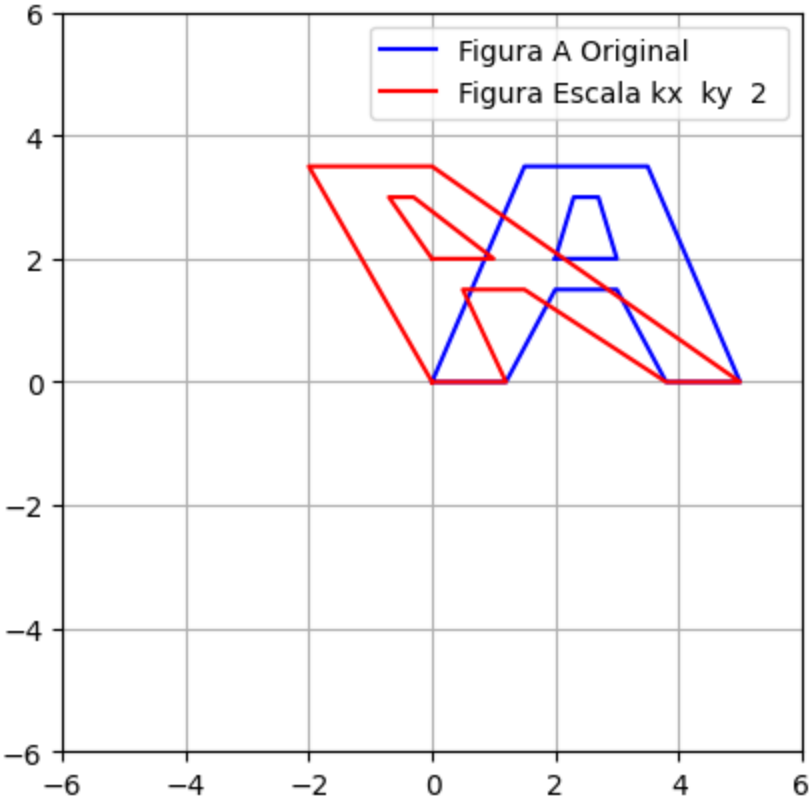
```
    (2,1.5,1),
    (3,1.5,1),
    (3.8,0,1),
    (5,0,1),
    (3.5,3.5,1),
    (1.5,3.5,1),
    (0,0,1)
])
A1 = np.array([
    (2,2,1),
    (3,2,1),
    (2.7,3,1),
    (2.3,3,1),
    (2,2,1)
])

S = np.array([
    [2, 0, 0],
    [0, 2, 0],
    [0, 0, 1]
])

# Matriz de rotación con theta = pi/3
theta = 3 * np.pi/2
R = np.array([
    [np.cos(theta), np.sin(theta), 0],
    [-np.sin(theta), np.cos(theta), 0],
    [0, 0, 1]
])

# Aplicar la rotación a los puntos de A y A1
Ax, Ay = [], []
A1x, A1y = [], []

for row in A:
    output_row = S.dot(row)
    Ax.append(output_row[0])
    Ay.append(output_row[1])

for row in A1:
    output_row = S.dot(row)
    A1x.append(output_row[0])
    A1y.append(output_row[1])

# Graficar las figuras originales y transformadas
plt.plot(A[:, 0], A[:, 1], color="blue", label="Figura A Original")
plt.plot(A1[:, 0], A1[:, 1], color="blue")
plt.plot(Ax, Ay, color="red", label="Figura Escala kx  ky  2 ")
plt.plot(A1x, A1y, color="red")

# Configuración del gráfico
ax = plt.gca()
ax.set_xticks(np.arange(-6, 8, 2))
ax.set_yticks(np.arange(-6, 8, 2))
plt.gca().set_aspect('equal', adjustable='box')
plt.grid()
plt.legend()
plt.show()
```



In [86]:

```
import numpy as np
import matplotlib.pyplot as plt

# Puntos de las figuras
A = np.array([
    (0,0,1),
    (1.2,0,1),
    (2,1.5,1),
    (3,1.5,1),
    (3.8,0,1),
    (5,0,1),
    (3.5,3.5,1),
    (1.5,3.5,1),
    (0,0,1)
])
A1 = np.array([
    (2,2,1),
    (3,2,1),
    (2.7,3,1),
    (2.3,3,1),
    (2,2,1)
])

S = np.array([
    [2, 0, 0],
    [0, 2, 0],
    [0, 0, 1]
])
D = np.array([
    [1, -1, 0],
    [0, 1, 0],
    [0, 0, 1]
])

# Matriz de rotación con theta = pi/3
theta = 3 * np.pi/2
```

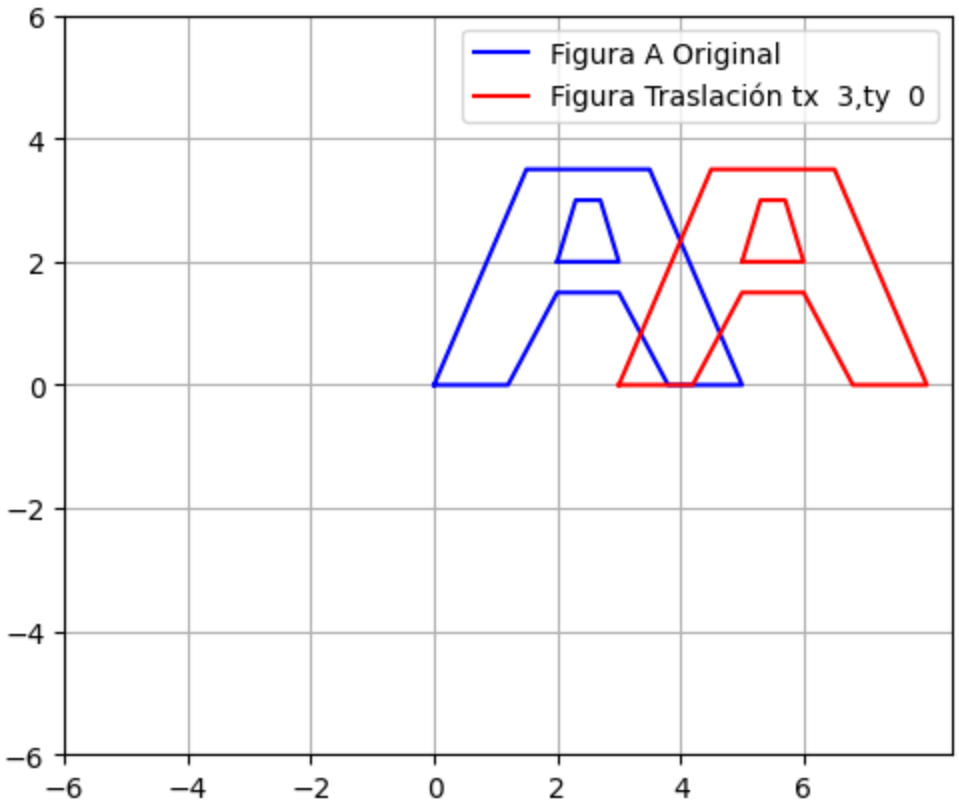
```
R = np.array([
    [np.cos(theta), np.sin(theta), 0],
    [-np.sin(theta), np.cos(theta), 0],
    [0, 0, 1]
])

# Aplicar la rotación a los puntos de A y A1
Ax, Ay = [], []
A1x, A1y = [], []

for row in A:
    output_row = D.dot(row)
    Ax.append(output_row[0])
    Ay.append(output_row[1])

for row in A1:
    output_row = D.dot(row)
    A1x.append(output_row[0])
    A1y.append(output_row[1])

# Graficar las figuras originales y transformadas
plt.plot(A[:, 0], A[:, 1], color="blue", label="Figura A Original")
plt.plot(A1[:, 0], A1[:, 1], color="blue")
plt.plot(Ax, Ay, color="red", label="Figura Escala kx  ky  2 ")
plt.plot(A1x, A1y, color="red")

# Configuración del gráfico
ax = plt.gca()
ax.set_xticks(np.arange(-6, 8, 2))
ax.set_yticks(np.arange(-6, 8, 2))
plt.gca().set_aspect('equal', adjustable='box')
plt.grid()
plt.legend()
plt.show()
```



In [88]:

```
import numpy as np
import matplotlib.pyplot as plt

# Puntos de las figuras
A = np.array([
    (0,0,1),
    (1.2,0,1),
    (2,1.5,1),
    (3,1.5,1),
    (3.8,0,1),
    (5,0,1),
    (3.5,3.5,1),
    (1.5,3.5,1),
    (0,0,1)
])
A1 = np.array([
    (2,2,1),
    (3,2,1),
    (2.7,3,1),
    (2.3,3,1),
    (2,2,1)
])

# Matriz de rotación con theta = pi/3
theta = 3 * np.pi/2
R = np.array([
    [np.cos(theta), np.sin(theta), 0],
    [-np.sin(theta), np.cos(theta), 0],
    [0, 0, 1]
])

# Aplicar la rotación a los puntos de A y A1
Ax, Ay = [], []
A1x, A1y = [], []
D = np.array([
    [1, -1, 0],
    [0, 1, 0],
    [0, 0, 1]
])
T = np.array([
    [1, 0, 3],
    [0, 1, 0],
    [0, 0, 1]
])

for row in A:
    output_row = T.dot(row)
    Ax.append(output_row[0])
    Ay.append(output_row[1])

for row in A1:
    output_row = T.dot(row)
    A1x.append(output_row[0])
    A1y.append(output_row[1])

# Graficar las figuras originales y transformadas
plt.plot(A[:, 0], A[:, 1], color="blue", label="Figura A Original")
plt.plot(A1[:, 0], A1[:, 1], color="blue")
plt.plot(Ax, Ay, color="red", label="Figura Traslación tx  3,ty  0")
plt.plot(A1x, A1y, color="red")
```

```
# Configuración del gráfico
ax = plt.gca()
ax.set_xticks(np.arange(-6, 8, 2))
ax.set_yticks(np.arange(-6, 8, 2))
plt.gca().set_aspect('equal', adjustable='box')
plt.grid()
plt.legend()
plt.show()
```



In [ ]: