```python
import time
import matplotlib.pyplot as plt  # Adicionado para gerar o gráfico

def lfsr_galois(bits):
    # Estado inicial com um número grande (dependendo do número de bits)
    estado_inicial = 0xACE1
    lfsr = estado_inicial
    periodo = 0
    numero_gerado = 0

    # Gerar o número pseudo-aleatório com o tamanho desejado
    for i in range(bits):
        lsb = lfsr & 1  # Obtém o bit menos significativo (bit de saída)
        lfsr >>= 1      # Desloca o registrador para a direita

        if lsb == 1:    # Se o bit de saída for 1
            lfsr ^= 0xB400  # Aplica a máscara de realimentação (tap positions)

        # Adiciona o bit gerado ao número final
        numero_gerado = (numero_gerado << 1) | lsb

    return numero_gerado

# Função para medir o tempo de execução
def medir_tempo_lfsr(bits, tentativas=10):
    tempos = []
    for _ in range(tentativas):
        inicio = time.time()
        lfsr_galois(bits)
        fim = time.time()
        tempos.append(fim - inicio)

    tempo_medio = sum(tempos) / len(tempos)
    return tempo_medio

# Testando para diferentes tamanhos de números
tamanhos = [40, 56, 80, 128, 168, 224, 256]
resultados = []

for tamanho in tamanhos:
    tempo_medio = medir_tempo_lfsr(tamanho)
    resultados.append((tamanho, tempo_medio))

# Exibir os resultados em uma tabela
print("Tabela de Resultados LFSR (configuração Galois)")
print("Tamanho do Número (bits) | Tempo Médio (segundos)")
for tamanho, tempo in resultados:
    print(f"{tamanho} | {tempo:.6f} segundos")

###### Explicação do Código ######

# Este código implementa um Registrador de Deslocamento com Realimentação Linear (LFSR) utilizando a configuração Galois, capaz de gerar números pseudo-aleatórios de tamanhos variados.

### Valor inicial (estado_inicial = 0xACE1) ###
# O registrador é inicializado com o valor hexadecimal 0xACE1, equivalente a 44257 em decimal e 1010110011100001 em binário. Este valor deve ser diferente de zero para garantir que o LFSR não entre em um estado estático.

```

```
### Máscara de realimentação (feedback mask = 0xB400) ###
# A máscara 0xB400 determina as posições dos bits (taps) que serão utilizadas no processo de realimentação. Essa escolha de máscara é típica para LFSRs de 16 bits, garantindo boas propriedades pseudo-aleatórias.

### Processo de geração ###
# A cada iteração, o bit menos significativo (LSB) é extraído.
# O registrador é deslocado uma posição para a direita.
# Caso o bit de saída seja 1, a máscara de realimentação é aplicada via operação XOR.
# O bit gerado é adicionado à esquerda do número pseudo-aleatório final (numero_gerado).

### Função de medição de tempo ###
# A função medir_tempo_lfsr avalia o tempo médio necessário para gerar números de diferentes tamanhos em bits. Para cada tamanho, são realizadas 10 execuções para calcular o tempo médio de geração.

### Tamanhos testados ###
# O algoritmo foi testado para números de 40, 56, 80, 128, 168, 224 e 256 bits. Tamanhos superiores (como 512, 1024, 2048 e 4096 bits) não foram incluídos devido à limitação da largura do estado do LFSR implementado (16 bits), o que impossibilita gerar sequências suficientemente longas sem repetição ou colapsos.

###### Referências Algoritmo ######
# Adapted from:
# https://en.wikipedia.org/wiki/Linear-feedback_shift_register
# All credits to the authors.

###### Geração de Gráfico ######
# Visualização da relação entre o tamanho do número (bits) e o tempo médio de execução.

# Extraindo dados para o gráfico
tamanhos_bits = [r[0] for r in resultados]
tempos_execucao = [r[1] for r in resultados]

# Criando o gráfico
plt.figure(figsize=(10,6))
plt.plot(tamanhos_bits, tempos_execucao, marker='o', linestyle='-', color='g')
plt.title('Tempo de Execução para Geração com LFSR (Configuração Galois)')
plt.xlabel('Tamanho do Número (bits)')
plt.ylabel('Tempo Médio de Execução (segundos)')
plt.grid(True)
plt.xticks(tamanhos_bits)
plt.tight_layout()
plt.show()
```