**Processo de Criação de um Periférico usando a API do QEMU**

**Logotipo, nome da empresa

Descrição gerada automaticamente**

Novembro, 2023

**1º: Criação de arquivos .C e .H**

Qualquer periférico, sob o ponto de vista do framework do QEMU, pode ser interpretado como um *object*, seguindo o *QOM –* *Qemu Object Model*. Assim, para a definição e criação de um novo periférico, primeiro deve-se definir o seu tipo e, a partir deste, instanciar um objeto integrado à máquina emulada.

Para o **arquivo .C**, a primeira operação importante é a criação de um *struct* que defina o periférico e suas variáveis, inicializando seus registradores, *buffers*, ponteiros de interrupção do sistema (*irqmp*) , ponteiros que representam seu *parente object* (de quem é derivado) e variáveis de manuseio das *Memory Regions.* Na figura abaixo, pode ser vista a definição do *struct* do periférico *grlib-apbuart*:

Texto

Descrição gerada automaticamente

É possível observar a inicialização de variáveis relacionadas ao funcionamento do periférico, junto às relacionadas ao framework e APIs do QEMU em si.

Após a definição do *struct,* são definidos alguns macros que facilitam a definição do periférico no framework. O conjunto desses macros pode ser definido somente pelo *OBJECT\_DECLARE\_SIMPLE\_TYPE(),* o qual recebe como argumento o nome do *struct* definido e o nome do tipo do objeto em *uppercase* e com termos separados por *underline*. O macro, para o código da *apbuart*, pode ser visualizado abaixo:



Após essas definições, podem ser inseridas funções de uso geral para o periférico em questão, para melhor organização do código. Dentre as funções presentes nos devices, destacam-se *device\_read()* e ­*device\_write()*, as quais são responsáveis pelo interfaceamento com a região de memória emulada do dispositivo.

Dessa forma, todos os periféricos apresentam estas funções e uma região de memória atrelada a elas. A definição das funções de *device\_read()* e *device\_write()* pode ser melhor observada, para o caso da *apbuart*, abaixo:

Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente

Para relacionar as funções definidas com uma *MemoryRegion,* estas são definidas em um *struct MemoryRegionOps*, o qual pode ser visualizado abaixo:

Texto

Descrição gerada automaticamente

Além de atrelar as funções *device\_read()* e *device\_write()*, também é definido o *endianness* do sistema, o qual, neste caso, vem informado por um macro.

Após esta análise, vem à tona a necessidade de criação de uma classe para o objeto que representará o periférico. Normalmente, a classe gerada é derivada de uma outra mais geral. No caso da *apbuart,* a classe definida para esta é derivada de uma classe geral *DEVICE\_CLASS*, a qual apresenta as funções *device\_realize()* e *device\_reset()* atreladas. A primeira função refere-se à chamada de algumas outras funções que fazem parte da inicialização e integração da *MemoryRegion*, interrupção e outros aspectos do device ao framework. A segunda refere-se a seu próprio *reset*, resetando seu estado e registradores. Abaixo, podem ser vistas as funções de *device\_realize()* e *device\_reset()* sob a ótica da *apbuart:*

*Texto

Descrição gerada automaticamente*

Vale ressaltar a não trivialidade da função *device\_realize()*, a qual, neste caso, inicia a *MemoryRegion* referente à *apbuart*, sua interrupção em relação ao processador e relaciona a *MemoryRegion* ao *bus* do sistema.

Como essas funções são declaradas para que a classe seja derivada da *DEVICE\_CLASS*, agora é necessária a chamada da função *device\_class\_init()*, a qual inicia a classe do próprio periférico sob a classe original. Para a *apbuart*:

Tela preta com letras brancas

Descrição gerada automaticamente

É notável a relação entre a classe declarada e as funções *device\_reset()* e *device\_realize()* definidas anteriormente. Vale ressaltar que, sendo a *apbuart* um dispositivo de saída do sistema, há também a definição de algumas propriedades relacionadas a esta.

Por fim, basta a inicialização do tipo referente ao objeto e classes criados, unindo-os no struct *TypeInfo*. Para a *apbuart*:

Texto

Descrição gerada automaticamente

É importante ressaltar que alguns dos *defines* são gerados pelo macro *OBJECT\_DECLARE\_SIMPLE\_TYPE()*, sendo outros definidos no arquivo .h.

Para a inicialização do tipo junto ao framework, basta a definição da função *device\_register\_types()* e a chamada do macro ­*type\_init()*. Para o caso da *apbuart*, tem-se:

Texto

Descrição gerada automaticamente

Essas primeiras etapas compreendem todas as implementações necessárias que devem ser feitas ao código .C do objeto. Quanto ao código .h, são definidos alguns *defines* para os nomes dos tipos dos periféricos. Para o caso dos periféricos derivados da *grlib*:

*Texto

Descrição gerada automaticamente*

**2º: Inicialização do objeto do periférico junto à máquina a ser chamada**

A etapa anterior referiu-se inteiramente à definição do tipo e classe do periférico, mas não à inicialização de um objeto que o represente na máquina a ser chamada.

Para isso, os objetos dos periféricos são inicializados nas funções *machine\_init()* das máquinas a serem emulados. Para o caso da *apbuart,* por exemplo, seu objeto é inicializado na função *leon3\_generic\_hw\_init()*.

Para diferentes tipos de periféricos, podem ser necessária a chamada de diversas funções de inicialização (envolvendo a inicialização da interrupção do periférico junto ao processador do sistema, canais de saída e etc.). No entanto, algumas funções, destacam-se perante as outras estando presentes na maioria das inicializações. Para o caso da *apbuart*:

Tela preta com letras brancas

Descrição gerada automaticamente

Dentre as funções presentes, as mais gerais para a maioria dos periféricos são: *qdev\_new()*, que cria um novo objeto com base no tipo informado; *sysbus\_realize\_and\_unref()*, que inicia o objeto como sendo do tipo “conectável” a um *bus* da máquina emulada; sysbus*\_mmio\_map()*, que mapeia a *MemoryRegion,* definida no tipo do periférico, junto à *MemoryRegion* da máquina a ser chamada.

Especificamente para a arquitetura *sparc* e para o processador *LEON3,* também há a chamada da função *grlib\_apb\_pnp\_add\_entry()*, que conecta o device ao *APB BUS* do processador do sistema.

**2º: Estabelecimento de diretivas para a compilação dos arquivos adicionados**

Nas etapas de *configure* e *make* dos arquivos do repositório do QEMU a fim da geração de um executável, o sistema “filtra” os arquivos e códigos a serem compilados, dependendo da arquitetura alvo escolhida.

Dessa maneira, os códigos .C e .h adicionados devem ser referenciados nas diretivas de compilação *Kconfig* e *meson.build*, arquivos presentes nas pastas onde o código .C pode ser inserido. Por exemplo, para a *apbuart:*

Texto

Descrição gerada automaticamente

A diretiva de compilação para a *apbuart* e outros dispositivos da *grlib* é *CONFIG\_GRLIB*, definida no arquivo *Kconfig* presente na mesma pasta do código da inicialização do *LEON3.*

*Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente*

Instrução condicional de compilação para alguns arquivos representando os tipos de alguns periféricos, dentre eles a *apbuart*. Esta instrução está presente no arquivo *meson.build* localizado na mesma pasta do arquivo .C que define o tipo da *apbuart*. Vale ressaltar que esta depende diretamente da diretiva inclusa no *Kconfig* mostrada anteriormente.