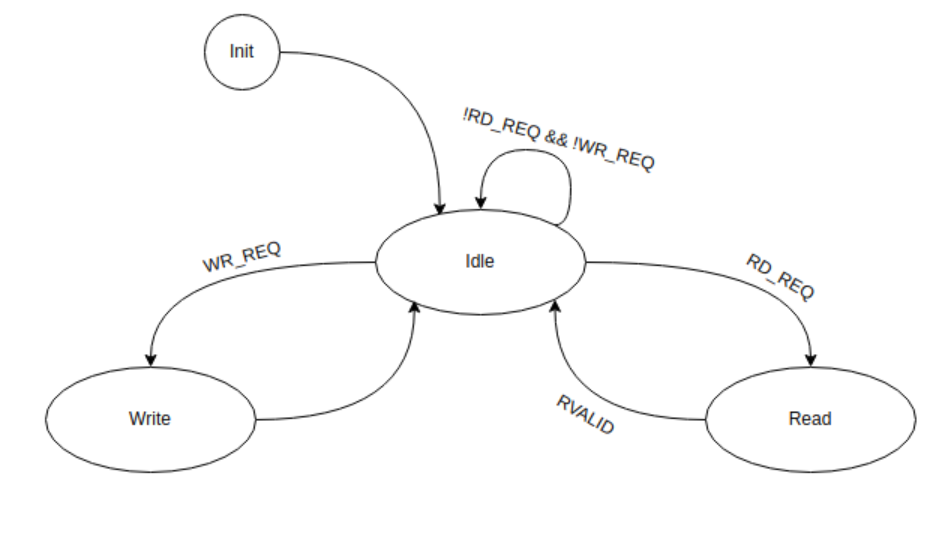
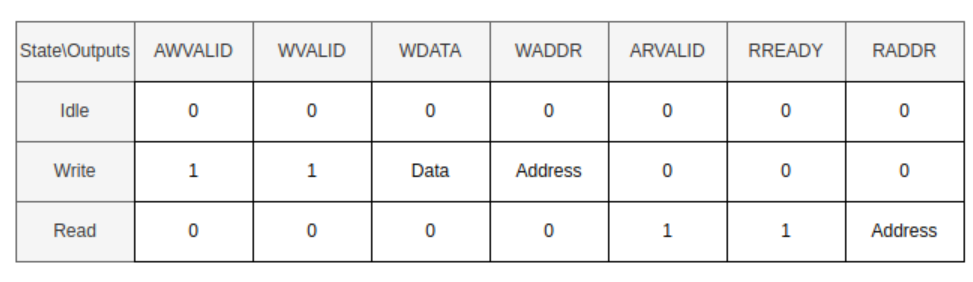
Em relação ao Master, desenvolveu-se totalmente o wrapper AXI, visto que a interface customizável que o Vivado gera possui muitos sinais que não são necessários neste projeto e para além disso, funciona como um exemplo, visto que possui uma máquina de estados que escreve em x registos e de seguida lê os mesmos endereços para verificar o correto funcionamento. Além disso, a interface master que o vivado gera não permite adicionar portos extras, pelo que era necessário adicionar no mesmo módulo uma interface slave para ultrapassar este problema.

Assim, a interface desenvolvida foi baseada na seguinte máquina de estados.

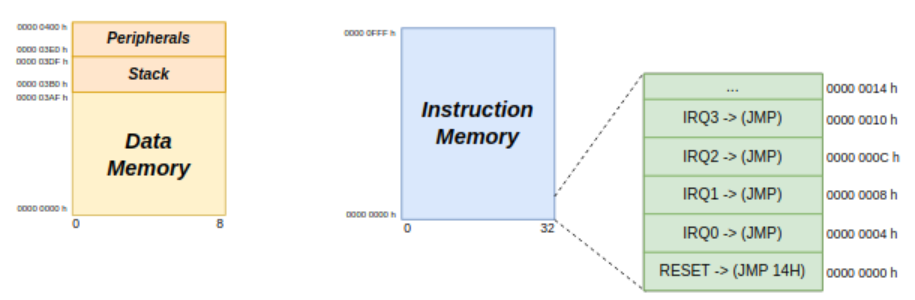


A seguir estão apresentados os outputs de cada estado.



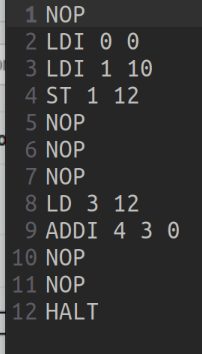
A memória de código tem 4k bytes e a memória de dados tem 1k bytes. Estas estão organizadas da seguinte forma.

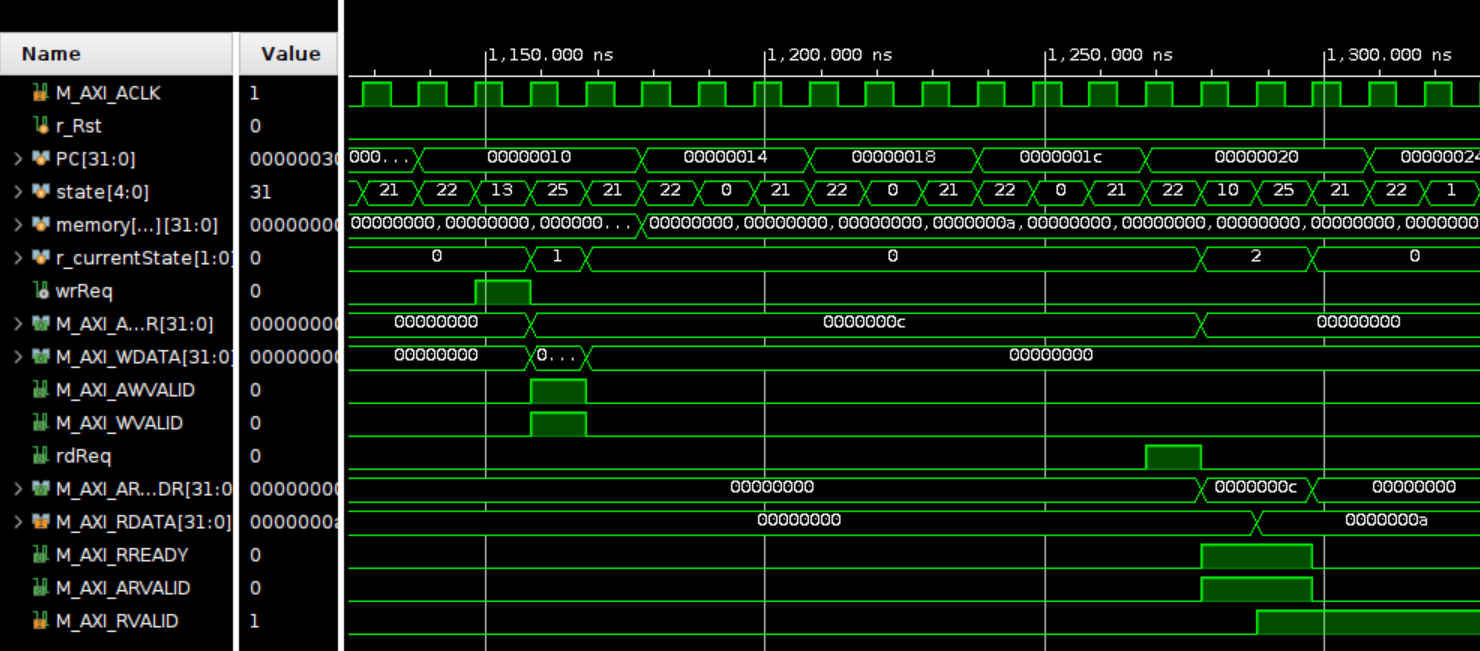
**ATENÇÃO:** A memória é alinha. Não está implementado a situação de lançar uma exceção. Ao testarem, se tentarem escrever/ler para um endereço que não seja múltiplo de 4 ele vai usar o endereço anterior múltiplo de 4.



**Test bench**

Neste momento o código que está na memória de código é o seguinte.



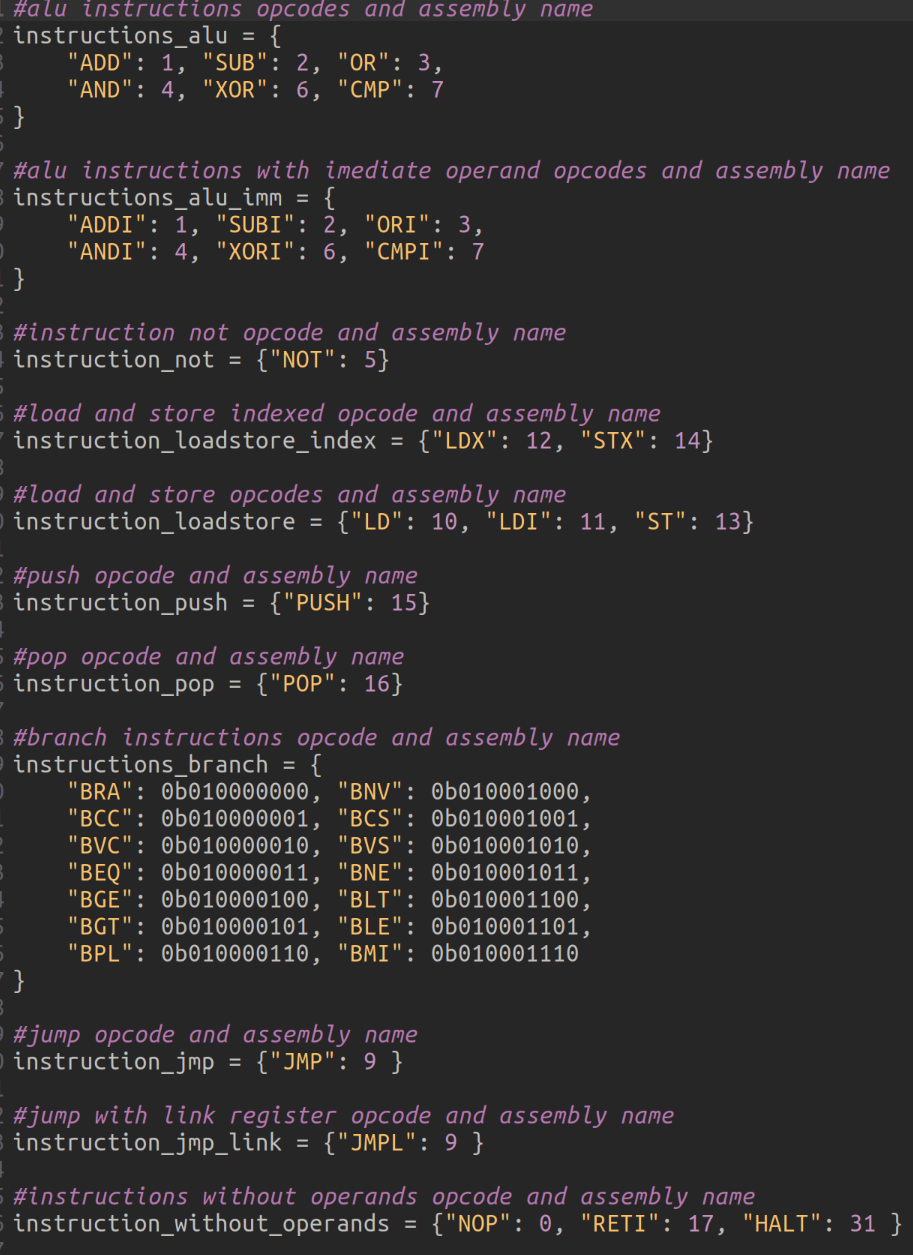


Neste testbench pode se ver que no estado 13 escreve-se o valor 0x0a no endereço 12 e mais à frente no estado 10 lê-se esse mesmo endereço e obtem–se o valor 0x0a.

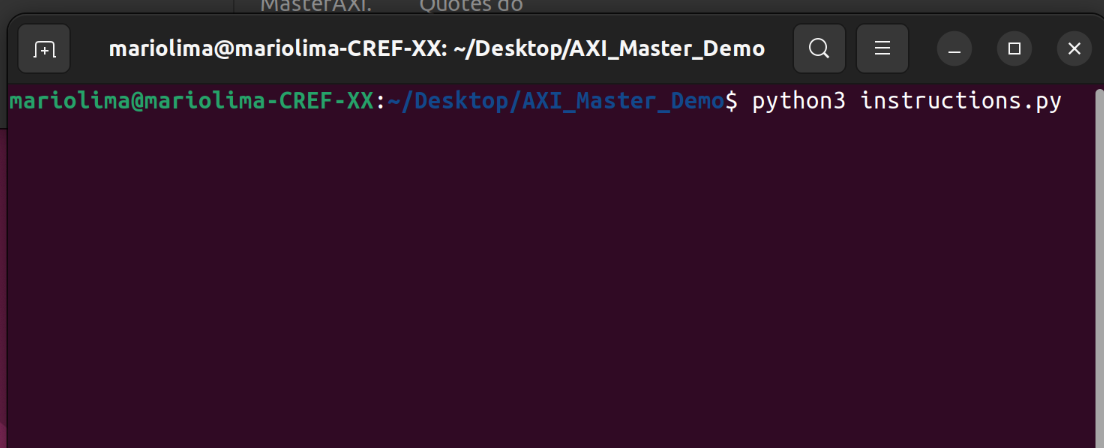
**Geração de código para teste**

Abrir o ficheiro code.txt

Escrever o código em assembly separado por espaços (e não terminar com espaços). As instruções assembly estão abaixo.



Executar o seguinte comando no diretório da pasta do projeto.



De seguida regenerar o IP da memória de código