ARQUITETURA DE COMPUTADORES

Laboratório 3 – Noção de Rotinas em Assembly e de métodos de passagem de parâmetros

Este laboratório destina-se a consolidar os conhecimentos de programação em assembly da arquitetura ARM. No laboratório será utilizado o emulador de um processador ARM designado de VisUAL. Este emulador implementa uma versão simplificada da *Unified Assembly Language (UAL). Contudo, no contexto do trabalho de laboratório, apenas o subconjunto de instruções indicado no guia de consulta rápida será suportado*. O emulador está disponível no link VisUAL, também disponível na página da cadeira. Para além da informação disponível na página da cadeira, deverá consultar o *User Guide* do emulador para realizar o laboratório.

O trabalho deve ser realizado fora do horário de laboratório, destinando-se este à demonstração e avaliação do trabalho realizado. No final da aula de laboratório deverá submeter o código Assembly no Fénix.

Para garantir a correção da solução, deverá validar a sua solução do laboratório no emulador (VisUAL) e confirmar os resultados, observando os valores finais dos registos ou o conteúdo da memória (Tools / View Memory Content). Pode igualmente colocar pontos de paragem no código (de tal forma que o emulador parará a execução sempre que atingir um destes pontos) clicando no número da linha do código.

Enunciado

1. Analise o programa "`L3.s`" e identifique as rotinas existentes, as suas funcionalidades e os métodos de passagem e parâmetros utilizados.
2. A rotina Pow efectua o cálculo da potência x^y, com x e y inteiros sem sinal:

   $$Pow(x,y) = x^y$$

   Para garantir a simplicidade do exercício, assumiu-se que o resultado desta rotina será sempre representado com 32-bits (i.e., menor que 4 294 967 296) pelo que domínio dos valores de x e y que garantem um resultado válido foi restrito (ver figura).

POW(x,y) = x^y		y											
		0	1	2	3	4	5	6	7	8	9	10	11
x	0	1	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	1	2	4	8	16	32	64	128	256	512	1024	2048
	3	1	3	9	27	81	243	729	2187	6561	19683	59049	177147
	4	1	4	16	64	256	1024	4096	16384	65536	262144	1048576	4194304
	5	1	5	25	125	625	3125	15625	78125	390625	1953125	9765625	48828125
	6	1	6	36	216	1296	7776	46656	279936	1679616	10077696	60466176	362797056
	7	1	7	49	343	2401	16807	117649	823543	5764801	40353607	282475249	1977326743
	8	1	8	64	512	4096	32768	262144	2097152	16777216	134217728	1073741824	8589934592
	9	1	9	81	729	6561	59049	531441	4782969	43046721	387420489	3486784401	31381059609
	10	1	10	100	1000	10000	100000	1000000	1E+07	1E+08	1E+09	10000000000	1E+11
	11	1	11	121	1331	14641	161051	1771561	1,9E+07	2,14E+08	2,358E+09	25937424601	2,85312E+11

   Analise o programa e identifique o método de passagem de parâmetros (entrada e saída) utilizados na rotina POW.

3. Verifique o valor guardado no PC antes da instrução `BL Pow`. Verifique o valor guardado no LR depois desta instrução.
4. Modifique o programa de modo a que a passagem de parâmetros da rotina Pow seja feita pela pilha. Guarde o código desenvolvido, para que o possa mostrar ao docente durante o laboratório.
5. Apesar de funcionalmente correcta, a implementação apresentada não é a mais eficiente, dado que faz uso de (y-1) multiplicações para calcular o resultado de Pow(x,y):

   x^4 = x.x.x.x (3 multiplicações)

   x^5 = x.x.x.x.x (4 multiplicações)

   x^8 = x.x.x.x.x.x.x.x (7 multiplicações)

   Uma implementação mais eficiente desta função passa pelo uso dos produtos anteriores, de modo a reduzir o número total de multiplicações:

   x^4 = (x^2)^2 (2 multiplicações)

   x^5 = (x^2)^2.x (3 multiplicações)

   x^8 = (x^4)^2 = ((x^2)^2)^2 (3 multiplicações)

   Genericamente, pode-se definir o algoritmo alternativo como:

```
/*****************************************/
if y=0 then
    pow(x,y)=1
else
    h=floor(y/2)
    if (y>1 AND y%2=0) then
        pow(x,y)= pow(x,h)^2// y is even
    else
        pow(x,y)= x.pow(x,h)^2// y is odd
/*****************************************/
```

   Tomando como ponto de partida o programa da alínea anterior, reescreva a rotina `POW` de modo a que o cálculo do resultado seja efectuado de forma recursiva e fazendo uso do algoritmo melhorado.

   NOTA: A passagem de parâmetros deve ser feita pela pilha e a rotina chama-se a si própria (i.e., invocando internamente a instrução `BL Pow`), não devendo reter valores temporários em registos ou variáveis.

   Guarde o código desenvolvido, para que o possa mostrar ao docente durante o laboratório.

6. Execute o programa implementado na alínea anterior considerando os seguintes números inteiros: X=3 e Y=9. Desenhe o mapa de memória da pilha correspondente ao instante em que a rotina `POW` é invocada com o segundo parâmetro de entrada (Y) nulo.