UNIOESTE – Universidade Estadual do Oeste do Paraná
Curso de Ciência da Computação
Disciplina de Linguagens de Montagem
Prof. Newton Spolaôr

Trabalho – valor: 100 pontos

O prazo (*deadline*) de entrega é no sábado 01/03/2025, 23:59 hs (horário de Brasília). A resolução dos exercícios deve ser enviada anexada em *chat* privado com o professor no Microsoft Teams. Solicita-se que todos os arquivos da resolução sejam comprimidos (usando software Winrar ou similar) e enviados em um único anexo com o nome completo do(s) aluno(s) do grupo (exemplo: SUN_TZU_E_REI_HELLU.zip).

Observações:

- O trabalho é em grupo de um ou dois alunos (preferencialmente, de dois alunos).
- Em caso de identificação de plágio, a nota 0 será atribuída para todos os envolvidos.
- Cada dia de atraso na entrega implica na perda de 50 pontos.
- Cada questão vale 50 pontos.
- O professor está à disposição (por *chat*, email ou pessoalmente) para esclarecer dúvidas relacionadas ao trabalho.
- Para cada resposta de questão escrita com apoio de uma fonte (exemplo: sites, artigos, livros, *slide* da aula...), inclua referência para essa fonte no final da resposta. No caso de falta dessa referência, há perda de 10 pontos.
- Para cada exercício que envolver algum tipo de implementação, é necessário entregar todos os códigos desenvolvidos pelo grupo, nas linguagens de programação solicitadas.

1) [Estudo dirigido] Resolva as Tarefas 2, 3 e 4 do Estudo dirigido 1 solicitado à turma no 1º bimestre. Nesse processo, certifique-se em cada tarefa de que o Passo 4 use como base o código C original criado pelo grupo na mesma tarefa. Por exemplo, na Tarefa 2, Passo 4, deve-se usar como base o código C original criado no Passo 3 desta mesma tarefa. O grupo é livre para revisar e editar, se necessário, a resolução do Estudo dirigido 1 entregue anteriormente, pois o estudo dirigido que valerá nota é o que será entregue no presente trabalho.

2) [Implementação] Escrever um código C com Assembly *inline* para arquitetura x86 de 32 bits capaz de resolver cada um dos problemas que seguem. Após encerrar o código Assembly *inline*, finalize o programa C imprimindo na tela o resultado encontrado em cada problema.

    a. Após definir valores para as variáveis inteiras globais com sinal *a*, *b*, *c* e *x* no programa C, calcule em Assembly o resultado da equação: $y = ax^2 + bx + c$. Cada uma dessas variáveis é do tipo *int* (32 bits). O resultado a ser impresso no final do programa C é a variável global com sinal *y*, também definida como tipo *int*.
    b. Após definir valores para as variáveis inteiras globais com sinal *a*, *b* e *c* no programa C, calcule em Assembly o resultado da expressão lógica: *x = a XOR (c OR NOT b) AND a*. Cada uma dessas variáveis é do tipo *int*. O resultado a ser impresso no final do programa C é a variável global com sinal *x*, também do tipo *int*.
    c. Sem usar instruções Assembly de transferência de dados que realizam simultaneamente conversão de tamanhos (MOVZX, MOVSX e similares) nem instruções Assembly de conversão de tamanhos (CBW, CWD, CWDE, CDQ e similares), defina um valor para a variável inteira global *f* com sinal, do tipo *char* (8 bits), no programa C. Após, implemente em Assembly a seguinte atribuição: *g=f*. Considere que a variável *g* é uma variável global com sinal do tipo *long long int* (64 bits). O resultado a ser impresso no final do programa C é a variável *g*.

Dica: use instruções aritméticas e de deslocamento de bits para auxiliar a resolver o problema.

d. Defina valores para as variáveis inteiras globais com sinal *a* e *b* no programa C. Considere que ambas variáveis são do tipo *short int* (16 bits). Após, em Assembly, desloque os bits de *a* para a esquerda, de modo que (1) todos os zeros à esquerda do primeiro bit 1 de *a* sejam removidos e (2) o preenchimento à direita seja realizado com zeros. Em seguida, substitua por um todos os bits zero à esquerda do primeiro bit 1 de *b*. Finalmente, preencha a variável global com sinal *c* do tipo *int* de modo que a metade mais significativa de *c* seja ocupada pelos 16 bits de *a*, enquanto que a metade menos significativa de *c* seja ocupada pelos 16 bits de *b*. O resultado a ser impresso no final do programa C é a variável *c*.

e. Defina valores para as variáveis inteiras globais com sinal *a* e *b* do tipo *short int* no programa C. Ainda em C, defina um valor potência de 2 para a variável inteira global sem sinal *c* do tipo *unsigned char* (8 bits). Um exemplo de valor que *c* poderia assumir seria 8, pois $2^3=8$. Após, em Assembly, processe *a* para que sua sequência de bits seja alterada como segue – nota: os bits com valor X que estavam inicialmente em *a* devem ser mantidos depois do processamento:

Valor original de *a*	XXXX	XXXX	XXXX	XXXX
Valor novo de *a*	0X1X	0X1X	0X1X	0X1X

De modo similar, processe *b* para gerar uma nova sequência de bits como segue.

Valor original de *b*	XXXX	XXXX	XXXX	XXXX
Valor novo de *b*	0000	XXXX	XXXX	1111

Em seguida, realize a soma dos valores processados de *a* e *b*. O resultado da soma deve ser estendido para 32 bits, preservando o bit do sinal. Na próxima operação, desloque para a direita por *k* posições o resultado estendido de 32 bits, preservando o sinal. O valor de *k* deve ser dado pela potência de 2 correspondente à variável *c*. Por exemplo, se $c=8=2^3$, então o valor de *k*, descoberto automaticamente pelo código Assembly, será 3. O valor de 32 bits obtido após deslocamento dos bits, armazenado em uma variável global com sinal *d* do tipo *int*, deve ser impresso no final do programa C.