UNIVERSIDADE FEDERAL DE PELOTAS – UFPEL
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO (CDTec)CURSOS DE CIÊNCIA DA COMPUTAÇÃO E ENGENHARIA DE COMPUTAÇÃO
DISCIPLINA DE PROGRAMAÇÃO DE SISTEMAS

PROFs.: Me. ANDERSON PRIEBE FERRUGEM

CHECKPOINTS

O **CHECKPOINTS** SERÁ UMA APRESENTAÇÃO EM VÍDEO DO GRUPO COM TODOS PARTICIPANTES COM CÓDIGO DISPONIBILIZADO VIA GITHUB;
O ENVIO É FEITO APENAS POR UM COMPONENTE DO GRUPO;
A DURAÇÃO MÁXIMA DO VÍDEO DEVERÁ SER DE **10 MIN** COM **TOLERÂNCIA** DE **5 MIN. (10-15)** ;
A APRESENTAÇÃO DEVERÁ MOSTRAR:

1) INTERAÇÃO ENTRE OS COMPONENTES;
2)ARGUIÇÃO DO FUNCIONAMENTO E DAS TÉCNICAS USADAS.

A APRESENTAÇÃO NÃO DEVERÁ SER APENAS:

1) APRESENTAÇÃO DE SLIDES;
2) APRESENTAÇÕES INDIVIDUAIS DOS COMPONENTES DO GRUPO.

EM CASO DE DÚVIDAS SOBRE A APRESENTAÇÃO PROCUREM POSTAR NO E-AULAS (DESTA FORMA A RESPOSTA FICA DISPONÍVEL A TODOS).

TRABALHO FINAL

O **TRABALHO FINAL** SERÁ UMA APRESENTAÇÃO EM VÍDEO DO GRUPO COM TODOS PARTICIPANTES COM CÓDIGO DISPONIBILIZADO VIA GITHUB;
O ENVIO É FEITO APENAS POR UM COMPONENTE DO GRUPO;
A DURAÇÃO MÁXIMA DO VÍDEO DEVERÁ SER DE **20 MIN** COM **TOLERÂNCIA** DE **+/-5 MIN. (15-25)** ;
A APRESENTAÇÃO DEVERÁ MOSTRAR:

1) INTERAÇÃO ENTRE OS COMPONENTES;
2)ARGUIÇÃO DO FUNCIONAMENTO E DAS TÉCNICAS USADAS.

A APRESENTAÇÃO NÃO DEVERÁ SER APENAS:

1) APRESENTAÇÃO DE SLIDES;
2) APRESENTAÇÕES INDIVIDUAIS DOS COMPONENTES DO GRUPO.

EM CASO DE DÚVIDAS SOBRE A APRESENTAÇÃO PROCUREM POSTAR NO E-AULAS (DESTA FORMA A RESPOSTA FICA DISPONÍVEL A TODOS).

UNIVERSIDADE FEDERAL DE PELOTAS
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO
CIÊNCIA DA COMPUTAÇÃO & ENGENHARIA DE COMPUTAÇÃO

Disciplina: PROGRAMAÇÃO DE SISTEMAS
TRABALHO PRÁTICO - Parte II

Projeto de um Sistema de Programação para um Computador Hipotético

1. Introdução

O trabalho que será descrito a seguir consiste em implementar um Sistema de Programação para o computador hipotético apresentado no Trabalho Prático - Parte I. Tal sistema será composto de quatro módulos que deverão operar de forma integrada: um **Montador** (de duas passagens), um **Processador de Macros** (de uma passagem, com definição e chamadas aninhadas), um **Ligador** e um **Carregador**.
Este trabalho deverá ser realizado por **Equipes** paralelas independentes, que desenvolverão instâncias diferentes do sistema**.** Cada equipe será coordenado por um **Líder**. Na formação dos grupos de trabalho, que é encargo da Equipe, devem ser observados o volume e o grau de dificuldade das atribuições.

O sistema deverá ser desenvolvido em **linguagem previamente escolhida em aula**, para plataforma Windows ou Linux.
A data limite para a entrega final será apresentada no e-aula. Com checkpoints intermediários para verificação do andamento do trabalho. A entrega deverá ser feita mediante **demonstração** prática por todos os componentes do grupo e apresentação de **documentação** formal sucinta do trabalho. O resultado do trabalho deverá ser entregue com toda a documentação (programas fontes, programa executável, documentação formal sucinta das estruturas de dados definidas, das funções desenvolvidas e estratégias adotadas) **até a data e hora limite pelo ambiente e-aula da disciplina**, e apresentado em vídeo, conforme solicitado pelo professor.

2. Descrição Geral

O objetivo geral dos módulos de software correspondentes ao Sistema de Programação a ser implementado é gerar programas que serão executados na Máquina Hipotética (Virtual) já implementada por algum componente da Equipe.

3. Descrição do Montador

Cada grupo ficará responsável pela implementação de um montador de duas passagens de acordo com as seguintes especificações:

3.1. Entrada do Montador

A entrada consistirá em um arquivo (programa) fonte, elaborado por qualquer editor de textos em formato texto, que será pré-processado pelo Processador de Macros e liberado para montagem sob o nome **MASMAPRG.ASM**. Os elementos que irão compor a linha serão separados por espaços em branco (no mínimo um). A linha deve conter, no máximo, 80 caracteres e obedecerá o seguinte formato:

```
[ [<label>] <opcode> [<operand1> [<operand2>]]] [<comentário>]
```

Os colchetes [ ] indicam campos opcionais. O campo <comentário> pode iniciar na coluna 1 desde que iniciado por um caractere ***** (neste caso ocupará toda linha). O <label> deve estar posicionado obrigatoriamente na primeira coluna, enquanto os demais campos podem ser colocados em qualquer posição na linha (exceto na primeira coluna).

3.2. Saída do Montador

O montador deverá fornecer como saída os seguintes dois arquivos :

<nome>.OBJ - código objeto resultante do procedimento de montagem conforme descrição da entrada para o ligador;

<nome>.LST : listagem do programa fonte e de seu código objeto.

O formato para o arquivo de listagem é (usar tamanhos fixos para os primeiros três campos):

[<endereço> <código gerado>] <linha> <linha do programa fonte>

No final do arquivo .LST deverá existir uma informação sobre o êxito do procedimento de montagem. Se a montagem for bem sucedida deverá aparecer uma mensagem do tipo "nenhum erro detectado". Caso haja erros, para cada um deles deverá ser informado o tipo de erro e a linha em que o erro ocorreu.

3.3. Elementos suportados pela linguagem de montagem

A linguagem definida para implementação do trabalho deverá considerar os elementos a seguir apresentados.

3.3.1. Conjunto de instruções

As instruções de máquina a serem tratadas pelo montador estão descritas no Anexo I (cópia do Trabalho Prático - Parte I). As instruções do montador (diretivas de montagem ou pseudo-operações) a serem suportadas são **START**, **END**, **INTDEF**, **INTUSE**, **CONST**, **SPACE** e **STACK**. Uma linha com diretivas de montagem possui o seguinte formato genérico:

[<label>] <diretiva> [<número/símbolo>] [<comentário>]

A pseudo-instrução START indica o endereço simbólico para início de execução de um programa (START *label*). Esse label é considerado como "nome do programa". A pseudo-instrução END simplesmente indica o fim do programa fonte. A diretiva denominada STACK terá a função de indicar o tamanho máximo de pilha a ser alocada para o módulo.
A diretivas INTDEF e INTUSE permitem, respectivamente, que se utilize referências externas a endereços de um outro programa (*label* INTUSE *)* e que se defina endereços para acesso externo, isto é, referências a partir de outros programas (INTDEF *label*).
As instruções CONST e SPACE são definidas como no livro-texto.
O montador deverá passar para o ligador as instruções correspondentes a estas diretivas.

3.3.2. Símbolos (labels)

Os símbolos serão formados por caracteres alfanuméricos, sendo o primeiro obrigatoriamente alfabético. Um símbolo pode possuir no máximo 8 caracteres.

3.3.3. Números e literais

Os números podem ser expressos nos formatos decimal e hexadecimal. Os literais são precedidos pelo prefixo "**@**". Exemplos:

63 = nº decimal H'3F' = nº hexadecimal @35 = literal em decimal

3.3.4. Expressões e modos de endereçamentos

Só existe um tipo de expressão: a *expressão simples* composta por um *símbolo* ou um *número* ou um *literal*.

Os modos de endereçamento definidos são o *direto*, o *indireto* e o *imediato*. O modo de endereçamento indireto é indicado, em qualquer dos operandos, através de um sufixo "**,I**" e o modo imediato é indicado pelo prefixo "**#**", observadas as restrições indicadas no Anexo 1.

3.4. Mensagens de Erro

Segue abaixo uma lista das possíveis mensagens de erro a serem consideradas pelo montador:

* *Caracter inválido.*
Unidade sintática não reconhecida (caracter inválido em algum elemento da linha).

* *Linha muito longa.*
Não deve haver mais de 80 caracteres numa linha.

* *Dígito inválido.*
Presença de um caracter não reconhecido como dígito para a base que está a ser usada.

* *Espaço ou final de linha esperado.*
Delimitador de final de linha depois do último operando ou instrução não reconhecido como válido.

* *Valor fora dos limites.*
Constante muito longa para o tamanho de palavra do computador.

* *Erro de sintaxe.*
Falta ou excesso de operandos em instruções, ou labels mal formados.

* *Símbolo redefinido.*
Referência simbólica com definições múltiplas.

* *Símbolo não definido.*
Referência simbólica não definida.

* *Instrução inválida.*
Mnemônico não corresponde a nenhuma instrução do computador.

* *Falta diretiva END.*
Indicação da ausência de pseudo-instrução END.

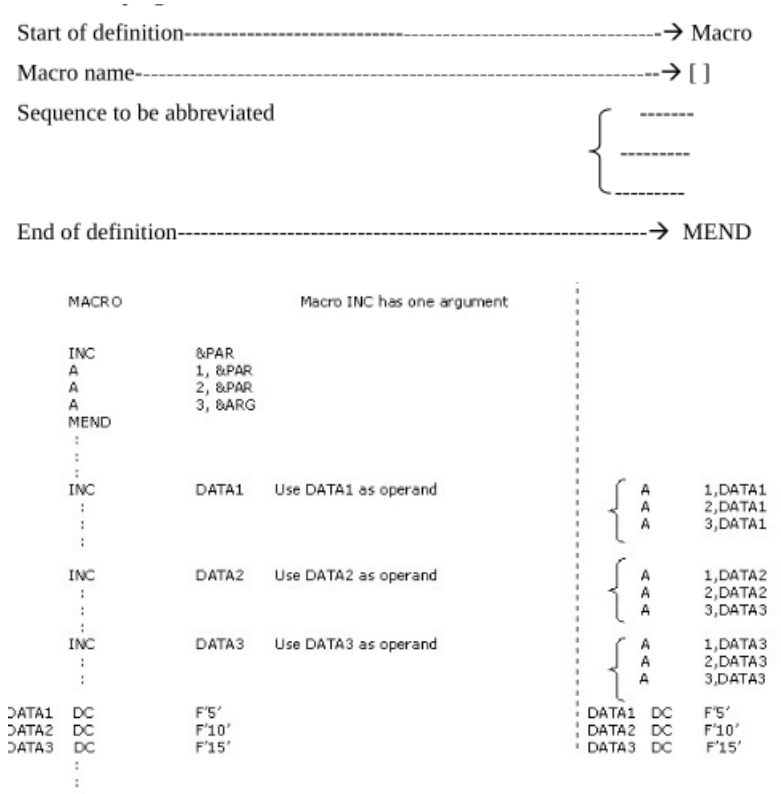
3.5. Interface com o usuário

Na chamada do montador será especificado o nome de arquivo fonte a ser traduzido.

O montador gerará dois arquivos de saída com o mesmo nome do arquivo fonte, porém um com o sufixo .OBJ e o outro com o sufixo .LST.

4. Descrição do Processador de Macros

O processamento de macros deve ser realizado antes da montagem, sendo ativado a partir do módulo principal integrador do macro-montador. Deve permitir a definição de macros dentro de macros (macros aninhadas), bem como a chamada de macros dentro de macros (chamadas aninhadas), sendo, portanto, implementado em uma só passagem. O programa receberá como entrada um arquivo fonte informado para montagem e gerará como saída outro arquivo fonte com o nome **MASMAPRG.ASM**.
As macros são definidas através das pseudo-operações MACRO e MEND e a sintaxe é a descrita abaixo; tal sintaxe está exemplificada no Anexo 2 por um programa que deverá ser utilizado para teste do Processador de Macros.
Também deve ser prevista a opção de listagem das macros segundo formato a ser combinado com o professor. Esta listagem deverá também conter algumas estatísticas sobre o uso das macros.



5. Descrição do Ligador de Módulos Objetos

O Ligador deverá ser implementado em duas passagens. Além da ligação, também executará a relocação completa de endereços, quando exigido **ligador-relocador**, considerando-se que o endereço de carga pode ser conhecido previamente (dependência de um Carregador Absoluto). Quando for exigido apenas um **ligador**, a finalização da relocação será deixada para o momento da carga (Carregador Relocador).

Na chamada do ligador os parâmetros serão os nomes dos arquivos objetos a serem ligados. Sendo que o nome do primeiro arquivo indicado será o nome dado ao executável (módulo de carga) resultante: *nome***.HPX**.
O ligador terá que informar ao carregador o tamanho da pilha (soma dos tamanhos necessários para cada módulo ligado) e o mapa de relocação (quando necessário). Também deverá ser informado o endereço inicial para execução.
Devem ser detectados, pelo menos, os erros correspondentes a:

- Símbolo global não definido : XXXXXXXX [<nome do módulo>]
- Símbolo global já definido : XXXXXXXX [<módulo atual>/<módulo anterior>]

Decisões a serem tomadas pela Equipe: forma de armazenamento das tabelas e código; ordem das tabelas.

6. Descrição do Carregador

O Carregador de Programas deverá ser integrado com o Emulador da Máquina Virtual para constituir a plataforma de execução de programas HPX (executáveis). Deverá, portanto, implementar um Carregador Absoluto ou Carregador Relocador (conforme o caso), que será ativado via um comando:

RUN <executável> Chamar Carregador e Desviar para Execução

No ato da carga, o Carregador deverá alocar a pilha do sistema, conforme o tamanho da pilha indicado no módulo executável e as especificações da Máquina Virtual. Após fazer a carga (e relocação, se necessário), o Carregador ativará o Emulador da Máquina Virtual para processar o programa solicitado.

7. Observações

Dever usada o Simulador da Máquina Virtual (hipotética) implementado na primeira etapa.

8. Bibliografia

CALINGAERT, Peter. **ASSEMBLERS, COMPILERS, AND PROGRAM TRANSLATION.** Computer Science Press. Inc, 1979.

STALLINGS, Willian. **COMPUTER ORGANIZATION AND ARCHITECTURE**. 5.ed. New Jersey: Prentice Hall, 1999.

TANENBAUM, Andrew. **STRUCTURED COMPUTER ORGANIZATION.** 3.ed. New Jersey: Prentice Hall, 1990.

9. Deadlines:

Montador – Checkpoint: 11/10
Processador de Macro: Checkpoint:25/10
Ligador: Checkpoint:08/11

ENTREGA TRABALHO FINAL:
27/11

ANEXO 1 - TABELA DE INSTRUÇÕES

Mnemônico (Sugerido)	Cód. de Máq.*	Tam. da Instrução (palavras)	N° de Ope-randos	Ação (comentário)	Modos de endereçamento (D/In/Im)	Obser-vações
ADD	02	2	1	ACC←ACC + opd1	D/In/Im	(#)
BR	00	2	1	PC←opd1	D/In	
BRNEG	05	2	1	PC←opd1, se ACC < 0	D/In	
BRPOS	01	2	1	PC←opd1, se ACC >0	D/In	
BRZERO	04	2	1	PC←opd1, se ACC = 0	D/In	
CALL	15	2	1	desvio para sub-rotina opd1	D/In	(%)
COPY	13	3	2	opd1←opd2	opd1: D/In opd2: D/In/Im	(#)
DIVIDE	10	2	1	ACC ← ACC / opd1	D/In/Im	(#)
LOAD	03	2	1	ACC ← opd1	D/In/Im	(#)
MULT	14	2	1	ACC ← ACC * opd1	D/In/Im	(#)
READ	12	2	1	opd1 ← *input stream*	D/In	
RET	16	1	0	Retorno de subrotina	-	(%)
STOP	11	1	0	término (fim) de execuçao	-	
STORE	07	2	1	opd1 ← ACC	D/In	
SUB	06	2	1	ACC ← ACC - opd1	D/In/Im	(#)
WRITE	08	2	1	*Output stream* ← opd1	D/In/Im	(#)
CONST	-	1	1	Declara constante	-	
END	-	-	0	Declara fim de módulo	-	
EXTDEF	-	-	1	Declara símbolo global	-	
EXTR	-	-	0	Label é símbolo externo	-	
SPACE	-	1	0	Reserva memória (palavra)	-	
STACK	-	-	1	Define pilha (tamanho máx.)	-	
START	-	-	1	Define início de execução	-	

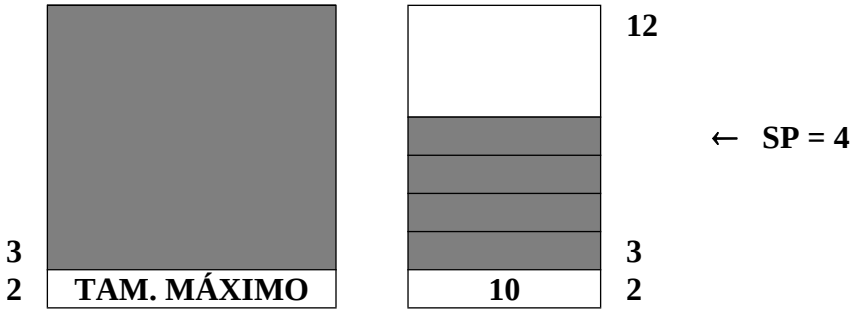
Legenda

* O código de máquina é alterável pelo modo de endereçamento indireto ou imediato.

(#) Instruções que podem ter endereçamento imediato.

% As instruções CALL e RET utilizam uma pilha para tratamento dos endereços de retorno, conforme descrito no item "Pilha do Sistema".

A Pilha do Sistema está localizada no início da memória a partir do endereço 2, com tamanho máximo definido pelo programa, seu topo é armazenado em um registrador **SP** e sua estrutura é a seguinte:



ANEXO 2 - EXEMPLO DE PROGRAMA FONTE

O programa a seguir é um exemplo de manipulação de macros tal como deverá ser utilizado no trabalho:

```
          START     TESTE
*
          MACRO
          SCALE     &RP
          MACRO
          MULTSC    &A,&B,&C
          LOAD      &A
          MULT      &B
          SHIFTR    &RP
          STORE     &C
          MEND
          MACRO
          DIVSC     &A,&B,&C
          LOAD      &A
          DIV       &B
          SHIFTL    &RP
          STORE     &C
          MEND
          MEND
*
          MACRO
&LAB      DISCR     &A,&B,&C,&D
&LAB      MULTSC    &A,&C,TEMP1
          MULTSC    TEMP1,@4,TEMP1
          MULTSC    &A,&B,TEMP2
          SUB       TEMP1
          STORE     &D
          MEND
*
          READ      A
          READ      B
          READ      C
          SCALE     3
          DISCR     A,B,C,D
          WRITE     D
          STOP
*
A         SPACE
B         SPACE
C         SPACE
D         SPACE
TEMP1     SPACE
TEMP2     SPACE
*
          END
```