

# Trabalho 1: Montador para a arquitetura do computador IAS

Edson Borin e Rafael Auler

23 de agosto de 2012

## 1 Introdução

Como visto anteriormente, um montador é uma ferramenta que converte código em linguagem de montagem para código em linguagem de máquina. Neste trabalho, você irá implementar um montador para a linguagem de montagem do computador IAS que produz como resultado um mapa de memória para ser utilizado no simulador do IAS.

## 2 Especificação do Montador

O formato do arquivo de entrada para seu montador é um arquivo texto onde cada linha pode conter:

- [rotulo:] [instrucao] [@ comentário]; ou
- [rotulo:] [diretiva] [@ comentário]

Comandos entre “colchetes” podem ser omitidos, ou seja, são opcionais. Dessa forma, as seguintes linhas são válidas em um programa em linguagem de montagem:

---

```
1 vetor:
2 vetor: .word 10
3 vetor: .word 10 @ Comentário após diretiva
4
5 .word 10
6 .word 10 @ Comentário após diretiva
7 @ Comentário sozinho
8
9 vetor: ADD M(0x100)
10 vetor: ADD M(0x100) @ Comentário após instrução
11
12 ADD M(0x100)
```

---

## 2.1 Parâmetros de execução

O seu montador deve ser um programa escrito em C que será executado pela linha de comando em um ambiente Linux e deverá reconhecer 2 parâmetros. O primeiro é o nome do arquivo de entrada (um arquivo texto em linguagem de montagem) e o segundo é o nome do arquivo de saída (mapa de memória que deverá funcionar no simulador IAS). O executável do seu montador deverá se chamar “montador-ias”. Exemplo de uso do montador:

---

```
1 $ ./montador-ias entrada.s saida.hex
```

---

As próximas seções descrevem o formato dos comandos na linguagem de montagem:

## 2.2 Comentários

Comentários são cadeias de caracteres que servem para documentar ou anotar o código. Estas cadeias devem ser desprezadas durante a montagem do programa. Todo texto à direita de um caractere “@” (arroba) é considerado comentário.

## 2.3 Rótulos

Os rótulos são compostos por uma cadeia de caracteres alfanuméricos e “\_”. A cadeia deve ser terminada com o caractere “:” (dois pontos) e não pode ser iniciada com um número. Exemplos de rótulos válidos são:

---

```
1 varx:
2 var1:
3 var2:
4 _varX3:
```

---

Exemplos de rótulos inválidos são

---

```
1 varx::
2 :var1
3 1var2:
4 var
```

---

## 2.4 Diretivas

Todas as diretivas da linguagem de montagem do IAS começam com o caractere “.” (ponto). As diretivas podem ter um ou dois argumentos dependendo da diretiva. Os argumentos das diretivas podem ser do tipo:

- **CONSTANTE**: argumentos deste tipo são valores numéricos. Exemplos: 0x00, -1, 12.
- **CONSTANTE+**: argumentos deste tipo são valores numéricos não negativos (inclui o zero). Exemplos: 0x00, 12.

- **CONSTANTE\***: argumentos deste tipo são valores numéricos maiores do que zero. Exemplos de números deste tipo são: 0x01, 0x000e, 12, 0b0011.
- **ROTULO**: argumentos deste tipo são rótulos do programa sem o caractere “:”. Exemplos são: `laco`, `var_x`, e `var1`.
- **NOME**: argumentos deste tipo são cadeias de caracteres alfanuméricos e o caractere “\_”. A cadeia não pode começar com um número.

A Tabela 1 apresenta a sintaxe das diretivas de montagem e o tipo dos argumentos.

Sintaxe	Argumento 1	Argumento 2
<code>.org</code>	CONSTANTE+	
<code>.align</code>	CONSTANTE*	
<code>.wfill</code>	CONSTANTE*	ROTULO ou CONSTANTE
<code>.word</code>	ROTULO ou CONSTANTE	
<code>.set</code>	NOME	CONSTANTE

Tabela 1: Sintaxe das diretivas de montagem e os tipos dos argumentos.

A descrição do comportamento de cada uma das diretivas podem ser encontradas na apostila de programação do computador IAS ou nos *slides* das aulas.

## 2.5 Instruções

As instruções do programa seguem o seguinte formato:

- **mnemônico** [**endereço**]

O mnemônico da instrução informa ao montador qual é a instrução que deve ser gerada, o campo endereço (opcional) informa qual o endereço a ser colocado no campo endereço da instrução. Note que este campo é opcional, pois existem instruções que não usam o campo endereço (ex: `RSH`). Caso o campo endereço seja omitido, o montador deverá preencher o campo endereço com zero.

O formato do campo endereço é:

- **M(end)**

Sendo que **end** é uma cadeia de caracteres sem espaço que pode ser um número (ex: `0xB` ou `11`) ou um rótulo (ex: `laco`).

A Tabela 2 apresenta os mnemônicos válidos e as instruções que devem ser emitidas para cada um dos casos. O montador não deve diferenciar maiúsculas de minúsculas no momento de reconhecer um mnemônico.

## 2.6 Representação dos números presentes no código em linguagem de montagem

O montador deve ser capaz de ler números em diferentes bases. Os caracteres que identificam uma determinada base precedem o número. Se um número iniciar com os caracteres “0x”, o montador deve esperar um número na base

Mnemônico	Instrução a ser emitida
LOADMQ	LOAD MQ
LOADMQMem	LOAD MQ,M(X)
STOR	STOR M(X)
LOAD	LOAD M(X)
LOADNeg	LOAD -M(X)
LOADMod	LOAD  M(X)
JUMP	JUMP M(X,0:19) se X for o endereço de uma instrução à esquerda de uma palavra.
	JUMP M(X,20:39) se X for o endereço de uma instrução à direita de uma palavra.
JUMPPos	JUMP+M(X,0:19) se X for o endereço de uma instrução à esquerda de uma palavra.
	JUMP+M(X,20:39) se X for o endereço de uma instrução à direita de uma palavra.
ADD	ADD M(X)
ADDMod	ADD  M(X)
SUB	SUB M(X)
SUBMod	SUB  M(X)
MUL	MUL M(X)
DIV	DIV M(X)
LSH	LSH
RSH	RSH
STORAddr	STOR M(X,8:19) se X for o endereço de uma instrução à esquerda de uma palavra.
	STOR M(X,28:39) se X for o endereço de uma instrução à direita de uma palavra.

Tabela 2: Mnemônicos válidos e as instruções que devem ser emitidas para cada um dos casos.

hexadecimal e aceitar os caracteres “a”, “b”, “c”, “d”, “e” e “f” como algarismos válidos, além dos números de “0” a “9”. O montador não deve diferenciar letras maiúsculas e minúsculas durante o processamento de números. Se um número for precedido pelos caracteres “0b”, o montador deve esperar um número na base binária e levantar um erro caso um caractere diferente de “0” ou “1” seja usado. Finalmente, um número precedido por “0o” indica que os próximos caracteres correspondem a um número na base octal (de “0” a “7”).

Caso nenhum dos casos anteriores seja obedecido, o número deve ser reconhecido como utilizando a base decimal. Exemplos:

---

```

1 varw: .word 0x10    @ Constante 16
2 varx: .word 22      @ Constante 22
3 LOAD M(0b1011)     @ Carrega uma palavra da posição de memória 11
4 varz: .word 0o377   @ Constante 255

```

---