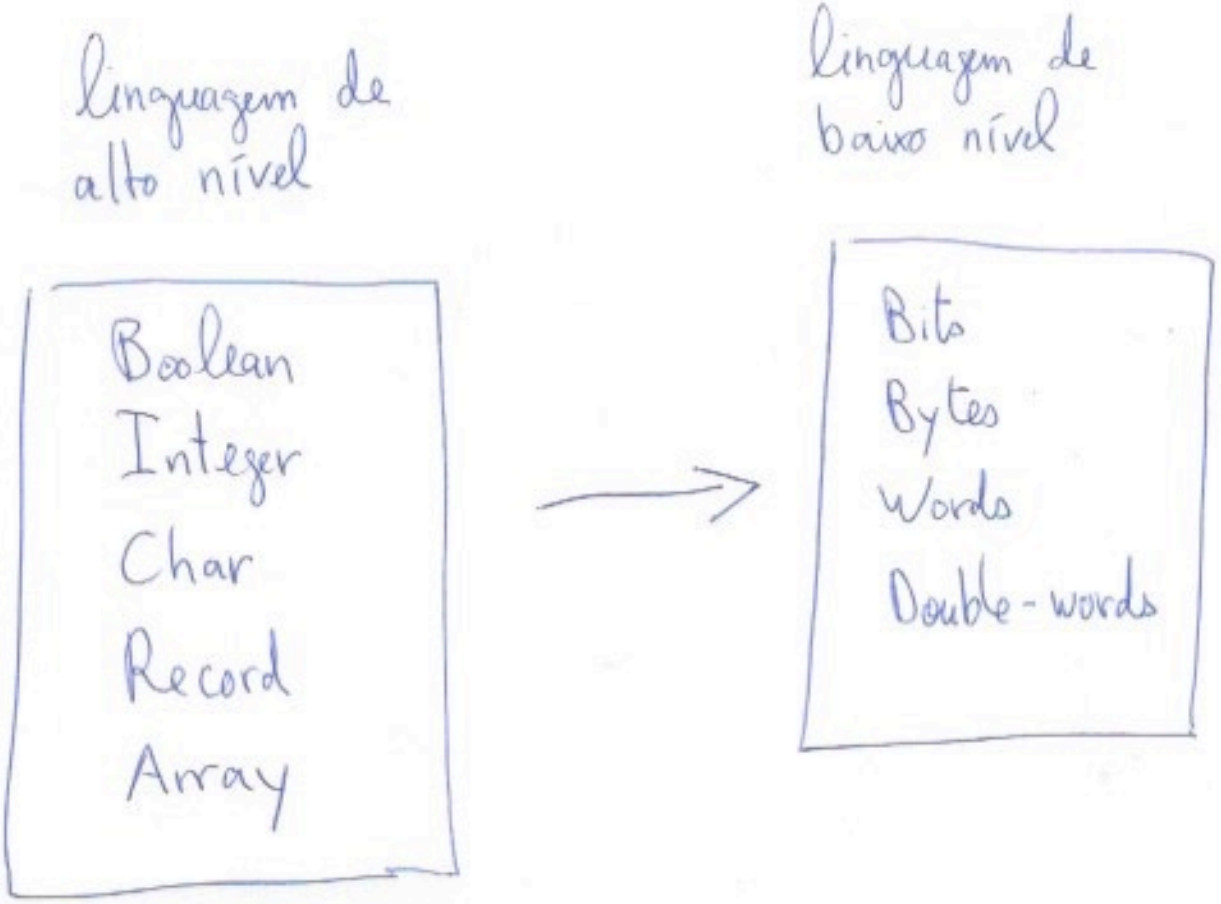
Ambiente de execução (Cap. 6 do livro PLPJ)

- Antes de fazermos o gerador de código, necessitamos de decidir como utilizar os recursos da máquina alvo (TAM, no nosso caso) para implementar a linguagem fonte (Triangle).

- Alguns aspectos essenciais:
  - Representação de tipos de dados
  - Avaliação de expressões
  - Alocação de memória para variáveis
  - Implementação de funções e procedimentos

Representação de dados



Princípios

1) Não pode haver ambiguidade: para um certo tipo de dados, valores distintos têm de ter representações distintas na máquina alvo.

2) Unicidade: um determinado valor tem sempre a mesma representação.

3) A representação de qualquer valor de um certo tipo de dados, ocupa sempre o mesmo espaço.

$\Rightarrow$ o compilador sabe exactamente quanto espaço deve alocar para cada tipo de variável.

Representação directa e indirecta

- Directa: normalmente é a representação em binário do valor.

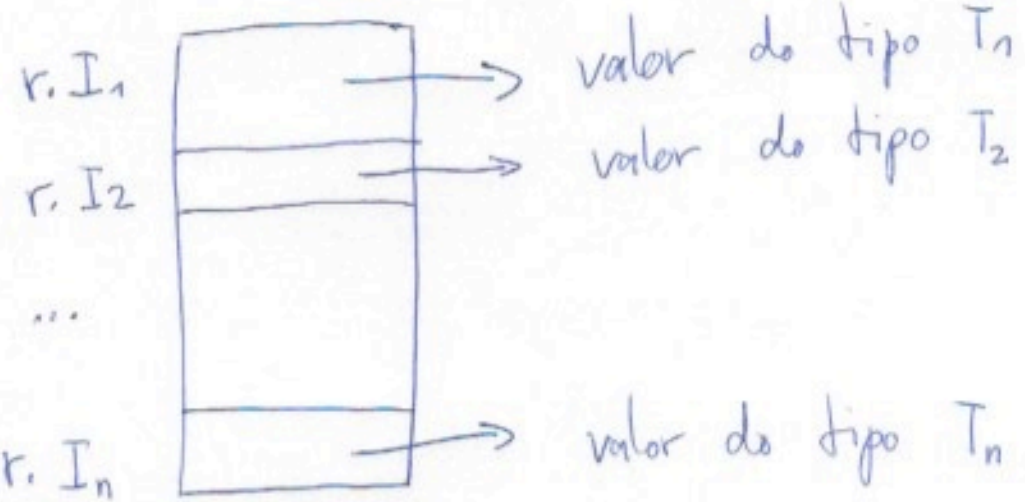
- Indirecta: temos um apontador que aponta para uma área de memória onde a representação do valor está.
  (essencial para alocação dinâmica de memória.)

Representação de tipos básicos na TAM

Tipo	Representação	Tamanho
Boolean	000...0 → False 000...1 → True	1 word
Char	repr. Unicode	1 word
Integer	complemento para 2	1 word ≀ 16 bits

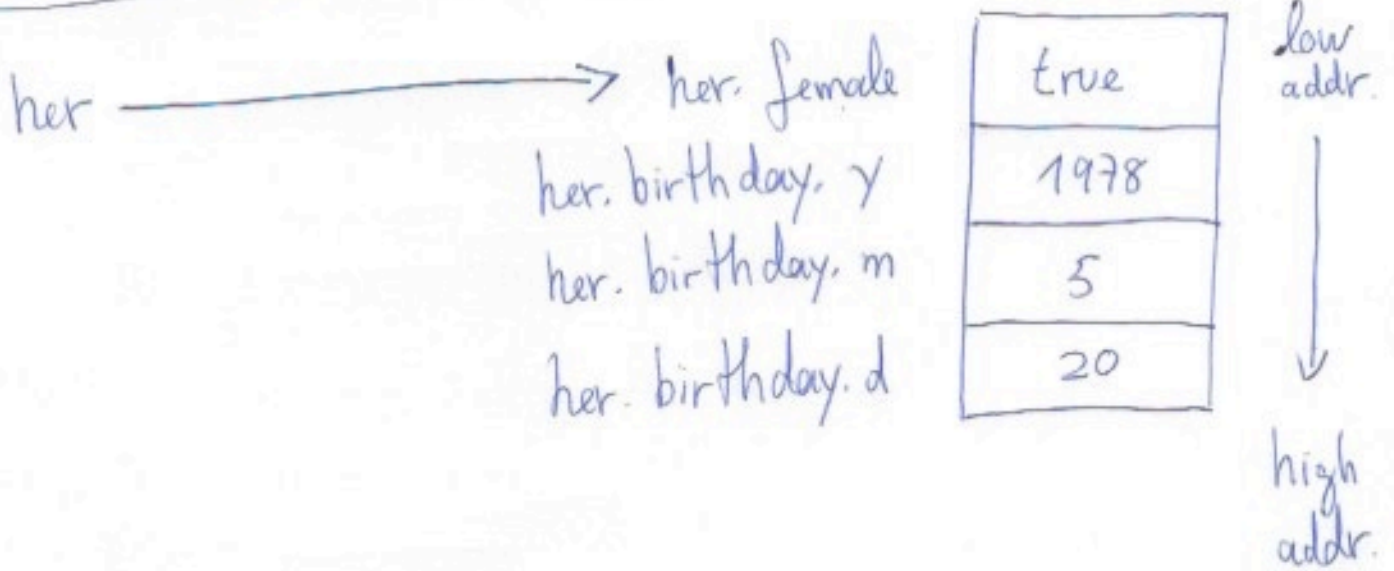
Representação de Records

- um Record é equivalente a uma struct em C.
- ocupa várias palavras contíguas de memória.



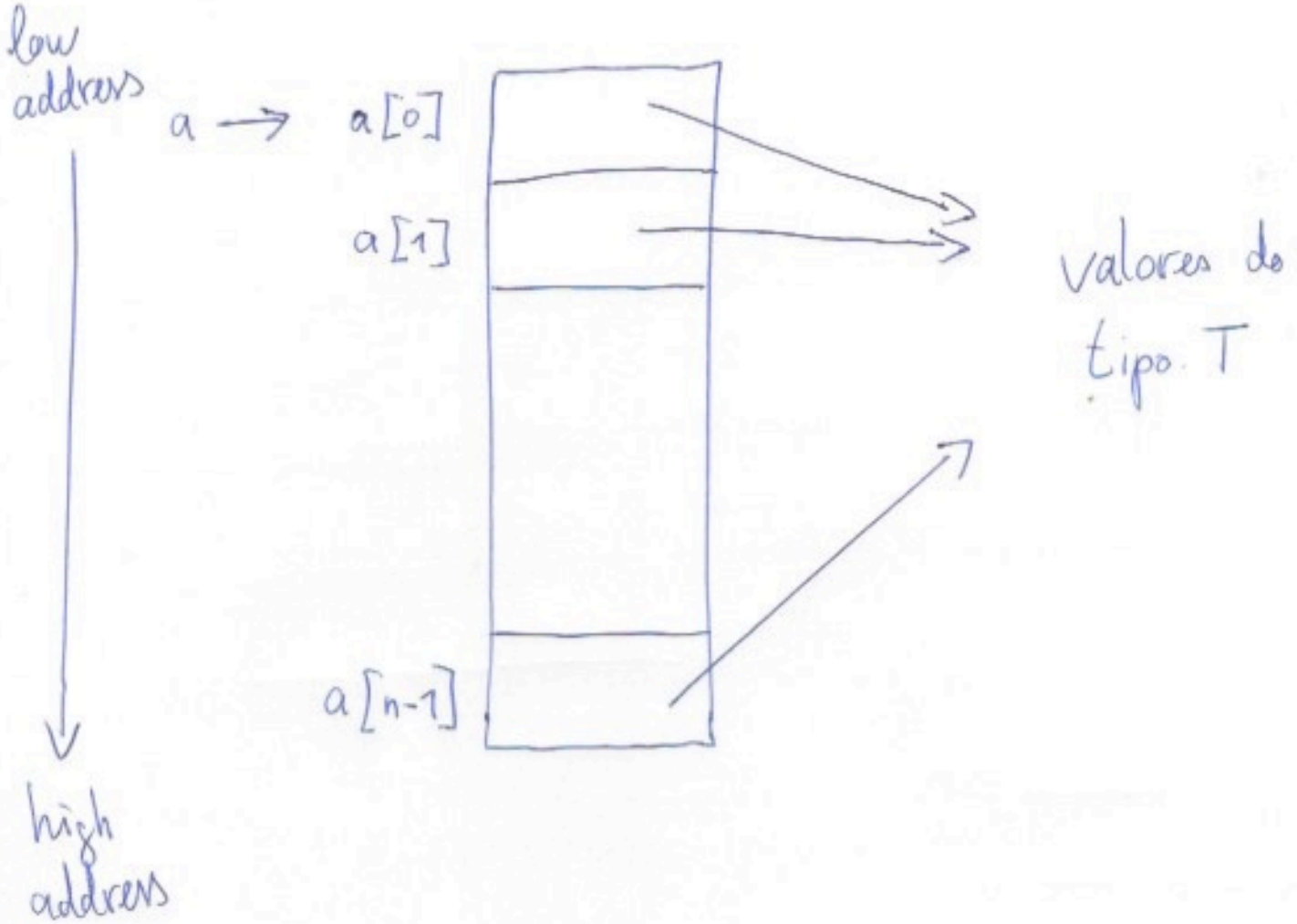
Exemplo

```
type Date = record
              y: Integer,
              m: Integer,
              d: Integer
            end;

type Details = record
                 female : Boolean;
                 birthday : Date
               end;

var her : Details;

her.female := true;
her.birthday.y := 1978;
her.birthday.m := 5;
her.birthday.d := 20;
```



Representação de arrays estáticos

- arrays estáticos : índices inferior e superior são conhecidos em tempo de compilação.
- ocupa palavras contíguas de memória.
- cada elemento ocupa o mesmo espaço.
  ⇒ permite indexar qualquer elemento em tempo constante.



Ver o cap. 6 do livro PLPJ para detalhes sobre representação de arrays dinâmicos e tipos de dados recursivos.

Avaliação de expressões

- Em máquinas de registos temos registos dedicados que podemos usar para guardar resultados temporários.

- Numa máquina de stack pura (como a TAM) esses registos não existem e usa-se o stack de execução para armazenar resultados temporários.

  — as expressões são avaliadas como se estivessem na notação pós fixa.

Ex: $a * b + (1 - (c * 2))$

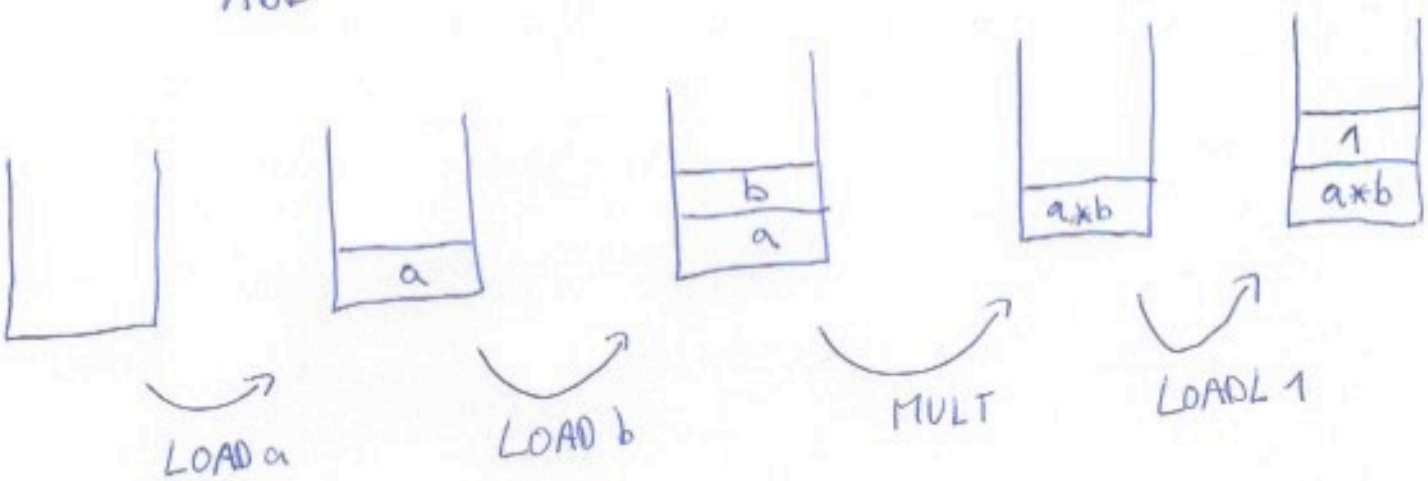
⇓

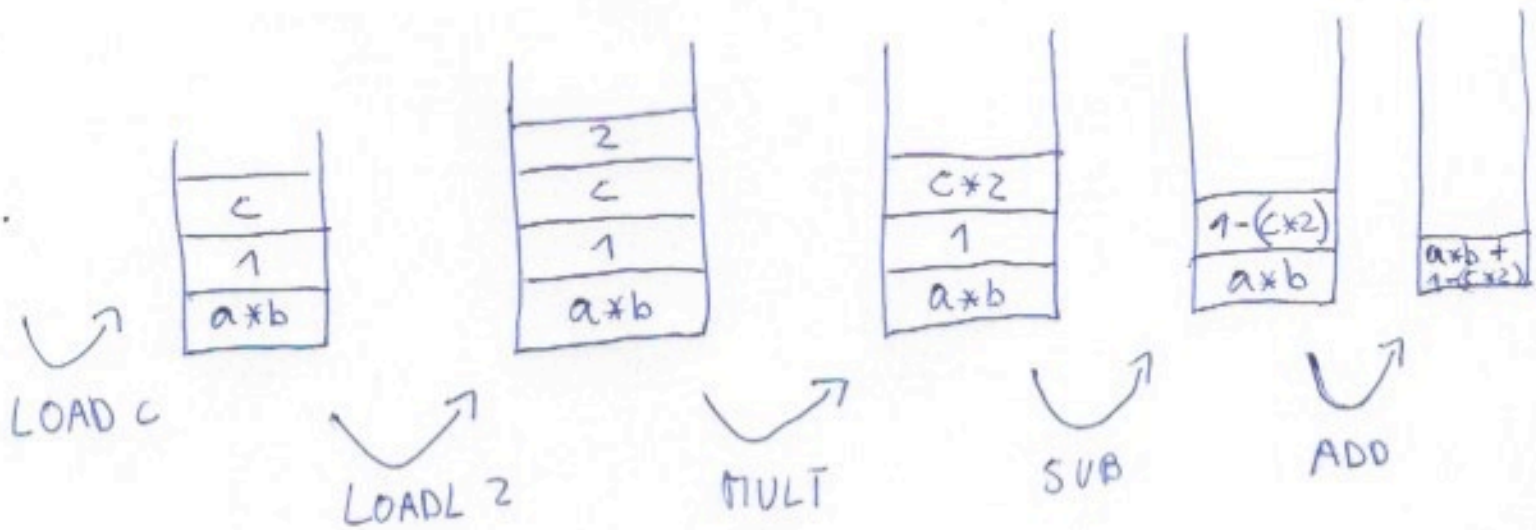
a b * 1 c 2 * - +

"pseudo-assemby" TAM:

```
LOAD a
LOAD b
MULT
LOADL 1
LOAD c
LOADL 2
MULT
SUB
ADD
```

A avaliação de expressões lógicas é feita de forma análoga.





Alocação de memória para variáveis globais

- O espaço é alocado de forma contígua, começando pela base do stack.
- Esse espaço chama-se global segment e fica reservado até a execução do programa terminar.

```
let
    var a: array 3 of Integer;
    var b: Boolean;
    var c: Char

in
    begin
        ...
    end
```