# Projecto ISA

- **Aspectos a considerar ao projectar ISA**

1. Que funcionalidades se pretende da ISA e do processador?

2. Plenitude

   Será que o conjunto de instruções contém todas as instruções necessárias para programar todas as tarefa pretendidas?

3. Ortogonalidade

   1. Instruções são ortogonais caso não sejam redundantes

   2. Um conjunto de instruções  bem projectado deve minimizar as redundâncias entre instruções, fornecendo deste modo ao programador apenas as funcionalidades pretendidas sob a forma de um número reduzido de instruções

4. Qual o número ideal de registos internos que permitam optimizar a ISA?

   O μP deve fornecer registos suficientes de modo a minimizar acessos à memória, melhorando deste modo o desempenho

# Projecto ISA

6. Quais os tipos e dimensões dos dados que o μP processará?

7. Será necessário garantir *backward compatibility* com outros μP?

8. As interrupções são necessárias?

   Mecanismos de *polling* podem não ser tão eficientes?

9. As instruções de salto condicionais são necessários?

   A ISA poderá ter que fornecer as *flags* (registos de 1-bit) para a memorização das várias condições

   **Como exemplo analisar o paragráfo 3.4 com a descrição do "*Relatively Simple Instruction Set Architecture*"**

# Projecto ISA: Resumo

## Ao projectar um ISA deve-se descrever:

1. O estado do processador e da memória

   Dimensões e números dos registos internos, organização da memória, estado de entrada/ saída se aplicável

2. Formato e interpretação dos dados nos registos: significado dos campos de registos

   Tipos de dados, formatos da instrução, interpretação do endereço efectivo

3. Interpretação da instrução: acções efectuadas por todas as instruções

   Especificação do ciclo *fetch-execute* e gestão das interrupções

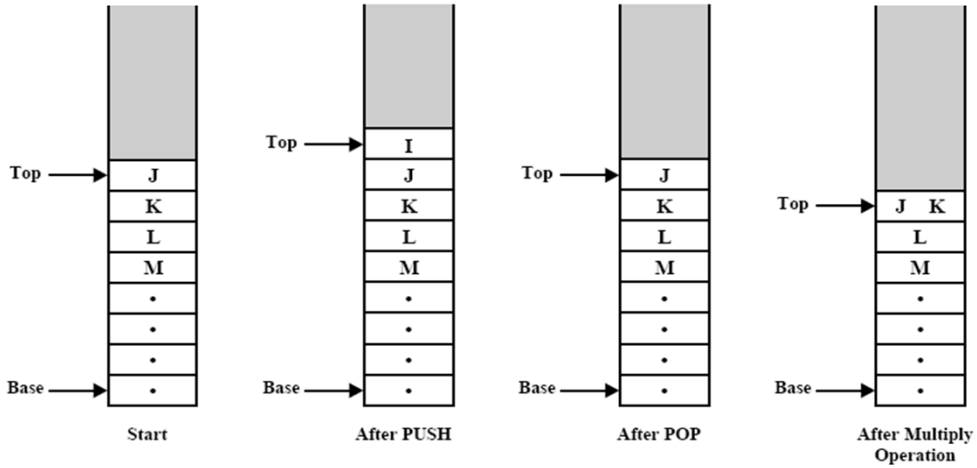
4. Execução da instrução: comportamento individual das instruções

   Especificando as classes de instruções em movimento de dados, salto, operações aritméticas e outros (para as instruções que não pertencem a nenhumas das classes anteriores), assim como acções efectuadas por cada instrução

# *Stack* - Pilha

- **Stack ( Pilha )**

  1. A pilha é um conjunto ordenado de elementos
  2. Num dado instante, apenas um elemento pode ser acedido
  3. O ponto de acesso é designado por topo da pilha
  4. É uma estrutura do tipo LIFO (*Last-In-First-Out*), em que os itens apenas podem ser inseridos (PUSH) ou removidos (POP) a partir do topo da pilha
  5. O comprimento da pilha é variável



# *Stack* - Pilha

- **Operações efectuadas sobre a pilha**

  1. PUSH

     Insere um novo elemento a partir do topo da pilha e actualiza o topo da pilha

  2. POP

     Remove o elemento presente no topo da pilha e actualiza o topo da pilha

  3. Operação de um único operando

     Efectua a operação sobre o elemento presente no topo da pilha e substitui o elemento do topo da pilha com o resultado

  4. Operação de dois operandos

     1. Efectua a operação sobre dois elementos do topo da pilha
     2. Remove os dois elementos do topo da pilha, actualizando o topo da pila
     3. Coloca o resultado no topo da pilha e actualiza o topo da pilha

  5. Qual o modo de endereçamento usado?

     Todas as operações referenciam uma única localização (o topo da pilha), pelo que o endereço do(s) operando(s) está/estão implícito na instrução

# *Stack* - Pilha

- **Implementações da pilha**
  1. Pilha acessível ao programador

     A ISA deve fornecer as operações como parte do conjunto de instruções

  2. Pilha apenas usado pelo CPU

     A ISA não fornece as operações como parte do conjunto de instruções, visto que estas operações apenas serão usadas na gestão das chamadas/retorno dos procedimentos/funções

  3. Um conjunto de localizações contíguas serão reservadas na memória principal ou memória virtual para armazenar os elementos da pilha

  4. Três endereços, normalmente, armazenados nos registos do CPU são necessários para garantir o funcionamento adequado das operações

     1. Apontador para o topo da pilha (*Stack pointer*)
     2. Endereço base da pilha (*Stack base*)
     3. Limite da pilha (*Stack limit*)

# *Stack* - Pilha

- **Implementações da pilha**
  1. Normalmente, a pilha cresce a partir dos endereços superiores para endereços inferiores

     Isto é, o limite é dado pelo endereço inferior e o topo da pilha é dado pelo endereço superior
  2. Normalmente, para acelerar operações sobre a pilha, costuma-se manter os dois elemento do topo da pilha em registos e apontar o topo da pilha para o terceiro elemento