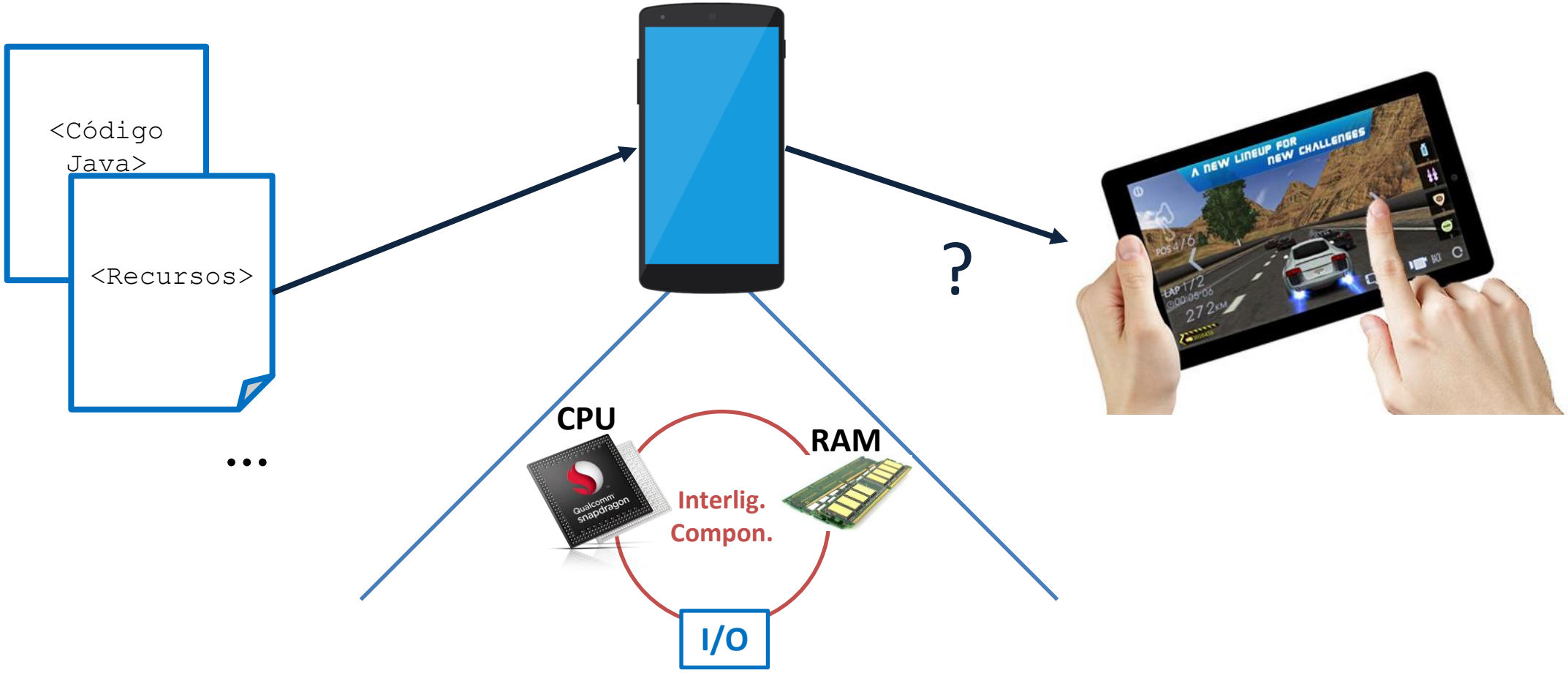
ANÁLISE DE EXECUÇÃO DE INSTRUÇÕES NUM CPU

TeSP de Aplicações Móveis

André Martins Pereira

IPCA
INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

EXECUÇÃO DE PROGRAMAS



EXECUÇÃO PELO PROCESSADOR

- O que já sabemos:
  - Processador comunica com outros componentes usando barramentos
  - CPU composto por Banco de Registos, ALU e Unidade de Controlo
  - Cada programa/aplicação em execução tem a si asssociado uma parte da memória principal (RAM)
  - 2 registo especiais:
    - IP – *Instruction Pointer* – Tem endereço de memória da próxima instrução
    - IR – *Instruction Register* – Tem valor (binário) da instrução que está a executar
  - 3 tipos de barramentos para comunicação memória <-> CPU:
    - Dados (*data bus*): responsável por transportar dados da memória para o processador, e vice-versa
    - Endereços (*address bus*): transporta valores de posições de memória
    - Controlo (*control bus*): Transporta valores RD ou WD para indicar que os valores transportados serão de leitura ou escrita, respetivamente
  - Vários tipos de instruções: aritméticas, saltos e envio de dados

EXECUÇÃO PELO PROCESSADOR

- O que falta saber: como é que o processador executa uma instrução
  - Que componente (ALU, banco registos, unidade controlo) faz o quê?
  - Que fases de execução existem?
  - O que acontece em cada uma delas?
  - O que muda antes e durante cada instrução?
  - E o que fazer depois de uma instrução executar?

???

EXECUÇÃO PELO PROCESSADOR

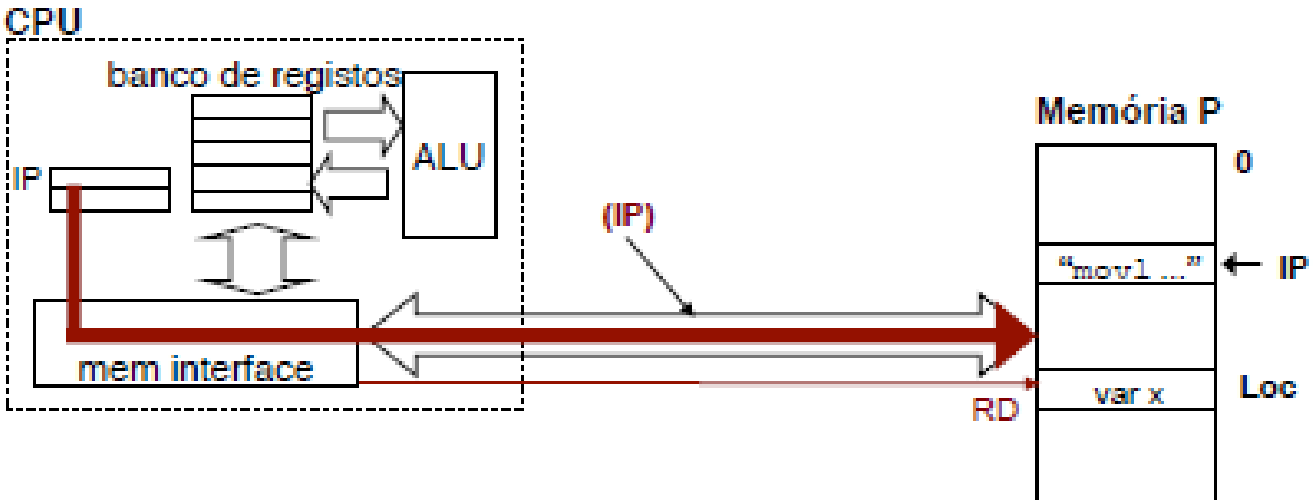
- 3 fases de execução:
  - *Fetch* (Procurar)
    - Leitura de uma instrução de memória
    - 3 etapas:
      - Colocar o valor do registo **IP** no barramento de endereços, pedir à memória o valor dessa posição
      - Colocar o valor obtido pela memória (que vem pelo barramento de dados) no **IR**
      - Atualizar o **IP** (incrementar)
  - *Decode* (Descodificar)
    - Analisar o padrão de *bits* lido de memória e…
    - Decidir o que tem de fazer
  - *Execute* (Executar)
    - Cálculo da localização dos operandos (e ir buscá-los, se necessário)
    - Executar a ação especificada
    - Guardar resultado, se necessário

EXECUÇÃO DE UMA INSTRUÇÃO

- Um exemplo prático: instrução ***movl end, %eax***
  - Esta instrução move o valor (*Long*) da posição ***end***para o registo ***%eax***
    - Usado para inicializar uma variável, por exemplo
  - ***end*** é um valor hexadecimal que representa o endereço de uma posição em memória
  - ***%eax*** é um registo guardado no banco de registos do processador

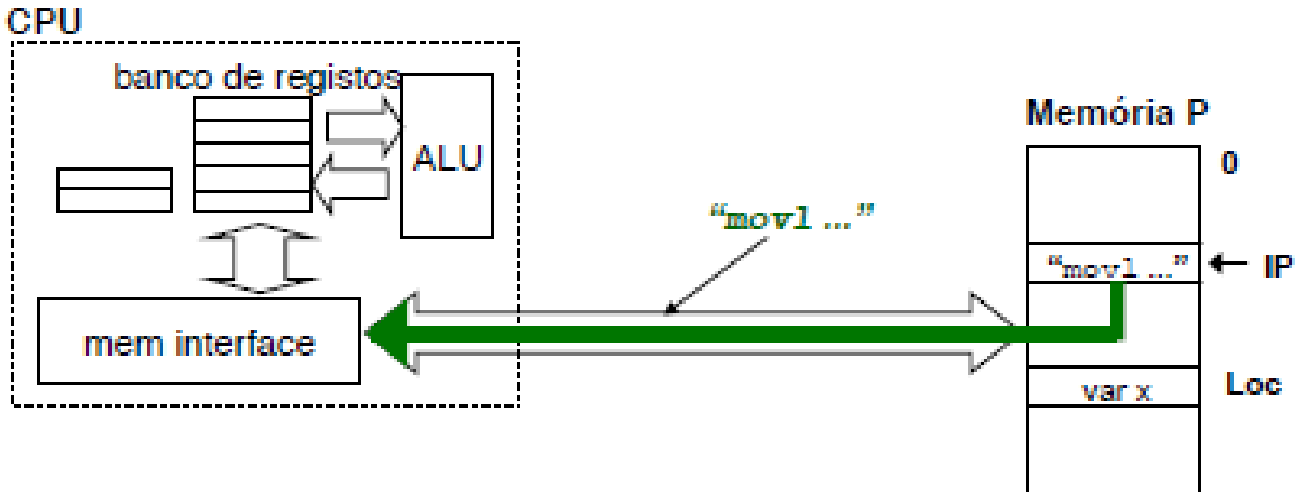
EXECUÇÃO DE UMA INSTRUÇÃO

- 1º passo: Ler a instrução de memória (*fetch*)
  1. CPU (unidade de controlo) consulta valor do **IP**
  2. Coloca-o no barramento de endereços (*address bus*)
  3. Coloca o sinal de controlo **RD** no barramento de controlo (*control bus*)
  4. Incrementa o **IP**



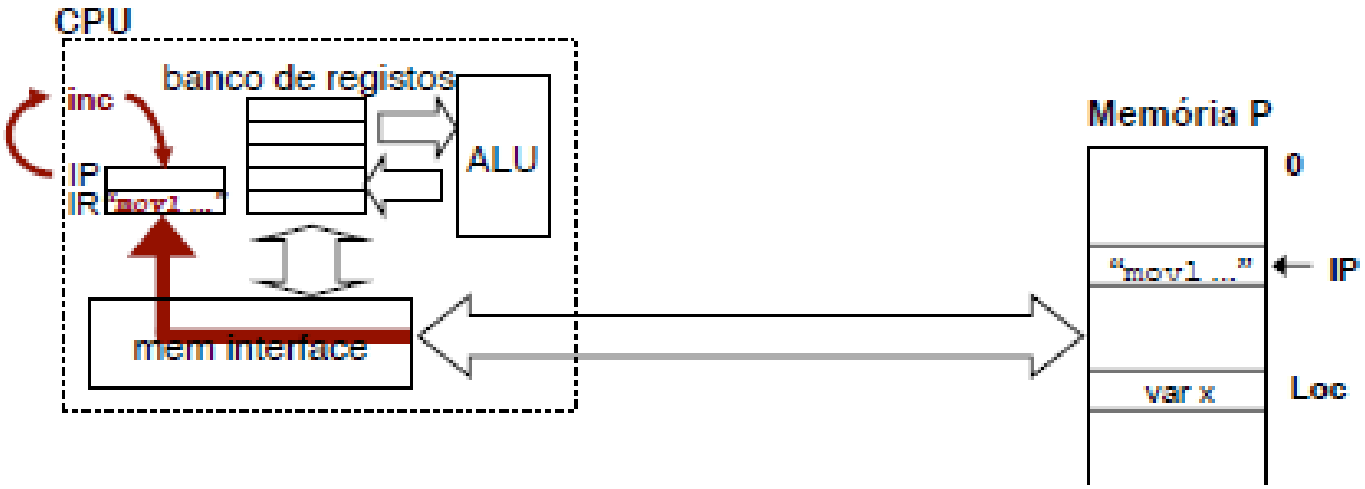
EXECUÇÃO DE UMA INSTRUÇÃO

- 1º passo: Ler a instrução de memória (*fetch*)
  5. A memória vai buscar a instrução ao endereço definido pelo **IP**
  6. Coloca-a no barramento de dados (*data bus*)



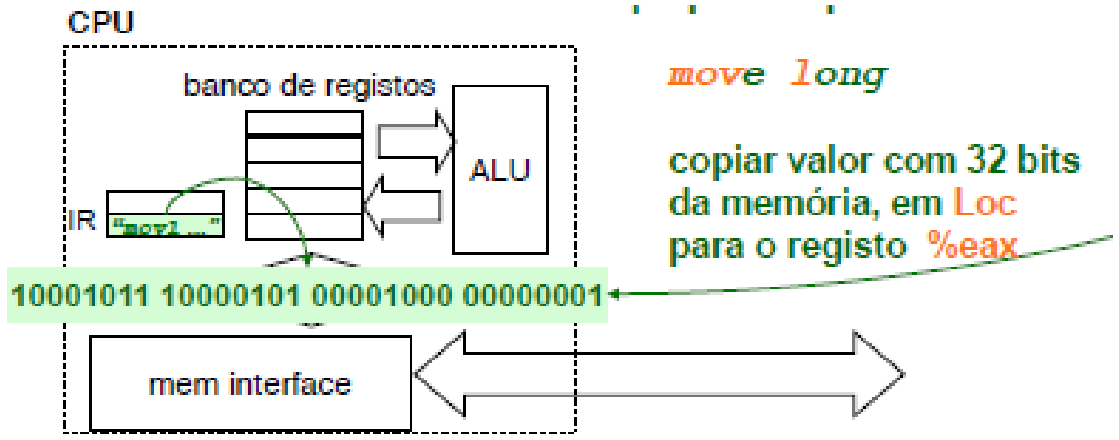
EXECUÇÃO DE UMA INSTRUÇÃO

- 1º passo: Ler a instrução de memória (*fetch*)
  7. CPU (unidade de controlo) lê a instrução do *data bus* e coloca-o no **IR**



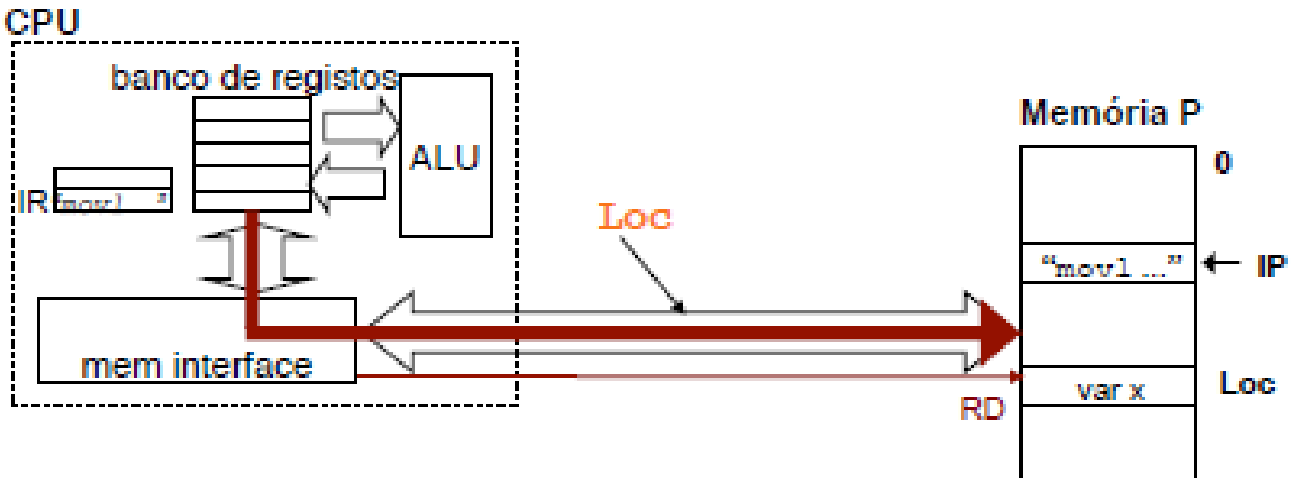
EXECUÇÃO DE UMA INSTRUÇÃO

- 2º passo: Descodificar a instrução (*decode*)
  1. Unidade de controlo do CPU interpreta o que está no IR
     - Percebe, ao ler os bits, que se trata de uma instrução ***move long***
     - Significa copiar valor de 32 bits do primeiro operando (***end***) para o segundo operando (***%eax***)
  2. Prepara-se para executar a operação



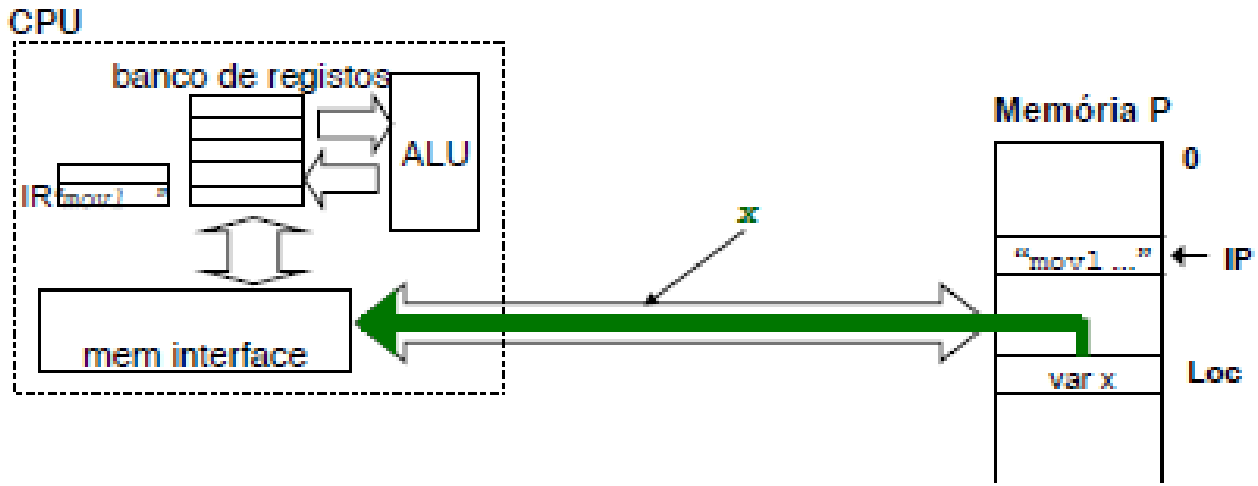
EXECUÇÃO DE UMA INSTRUÇÃO

- 3º passo: Executar a operação (*execution*)
  1. Unidade de controlo calcula o valor de ***end***(em binário) no *address bus*
  2. Envia o sinal **RD** para o *control bus*



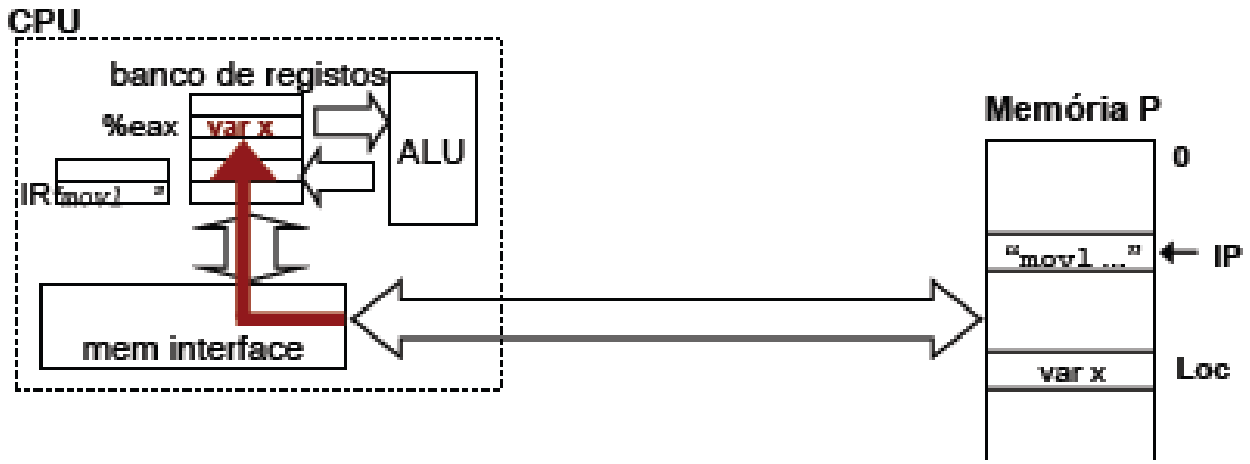
EXECUÇÃO DE UMA INSTRUÇÃO

- 3º passo: Executar a operação (*execution*)
  3. A memória vai ao endereço ***end*** buscar o valor pretendido
  4. Coloca-o na *data bus*



EXECUÇÃO DE UMA INSTRUÇÃO

- 3º passo: Executar a operação (*execution*)
  5. Unidade de controlo lê o valor do *data bus*
  6. Simplesmente coloca-o no registo ***%eax***



EXERCÍCIOS

- Com uma arquitetura de 32 bits e ordenação *Little Endian*, analisar as seguintes instruções:
  - `addl 0xC4, %eax` (16 bits em memória)
  - `movl %ebx, (0x8414)` (16 bits em memória)
  - `pushl %ebp` (8 bits em memória)
  - `incl %esp` (8 bits em memória)
  - `popl %ecx` (8 bits em memória)
- Assumindo os seguintes valores em registos/memória antes da execução de cada uma delas:
  - Registos
    - (IP) %eip = 0x8080
    - (IR) %eir = 0x10F1
    - (SP) %esp = 0x8414
    - %eax = 0x8408
    - %ebx = 0x88
  - Memória:
    - **De** `0x00008080` **até** `0x00008083: 0C 62 14 FF`
    - **De** `0x00008410` **até** `0x00008417: 09 10 2A 3B 4C 5D 6E 7F`

ANÁLISE DE EXECUÇÃO DE INSTRUÇÕES NUM CPU

IPCA
INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA