# The CPU IA-64: Key Features to Improve Performance

Ricardo Freitas
Universidade do Minho
22nd January 2004

## IA-64:

- Overview:
  - Speculation
  - Predication
  - EPIC – Explicit Parallel Instruction Computing
- Main Goal:
  - Registers Set
  - Register Stack Engine - RSE

## IA-64 Overview

Speculation:

- Data Speculation
- Control Speculation

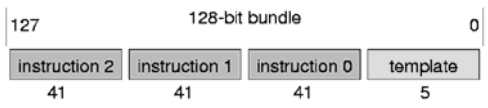
## Predication

To reduce branching 64 predicate registers (1 bit each) when predicate is false instruction is not executed

- **C code:**
  **r2=r1==0?r4+r5:r3+r6+1;**
- **IA-64 assembler:**
  **cmp.eq p1,p2=r0,r1;;** ← synchronization
  **(p1) add r2=r4,r5**
  **(p2) add r2=r3,r6,1**

# EPIC

- Explicit parallelism
  - Bundles of 3 instructions
  - Template field encodes
    - Type of execution units needed (M-unit for memory access, I-unit for integer operations, F-unit for floating point, B-unit for branching)
    - stop bit to express sequential dependency

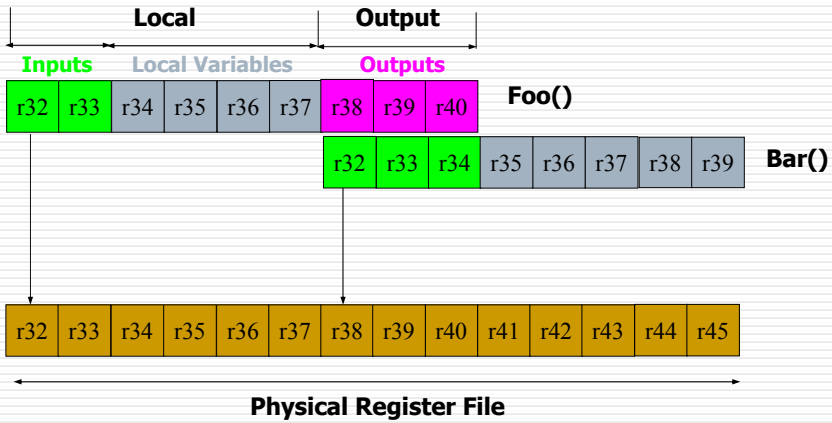



- Massive resources
  - 128 integer (64bits) & 128 floating point (82bits) registers
  - lots of execution units

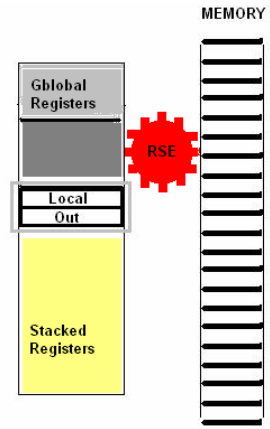
# Registers Set

- 128 general-purpose registers, each 64 bits wide.
  - Static general-purpose registers (r0 – r31)
  - Dynamic general-purpose registers (r32 – r127)
- 128 floating-point registers, each 82 bits wide, (f0 – f127).
- 8 branch registers, b0 through b7.

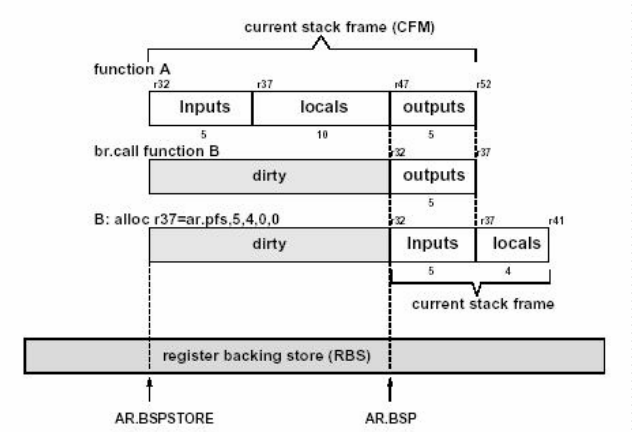
# Dynamic general-purpose registers




# Register Stack Engine



## Register Stack Engine/Re-Mapping




## Register Stack Engine

- Stack frame partition:
  - Local area: input parameters and local variables
  - Output area: output values
  - Can also be used as rotation registers

## **Register Remaping**

- Register Stacking
- Register Rotation

## Conclusions

- New instructions for register stack manipulation
- Large register file
- Automatic save/restore of general registers on function call/return
- Renaming scheme
- Transparent register stack spill/fill