Dec	Bin	Hex
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

FUNÇÕES

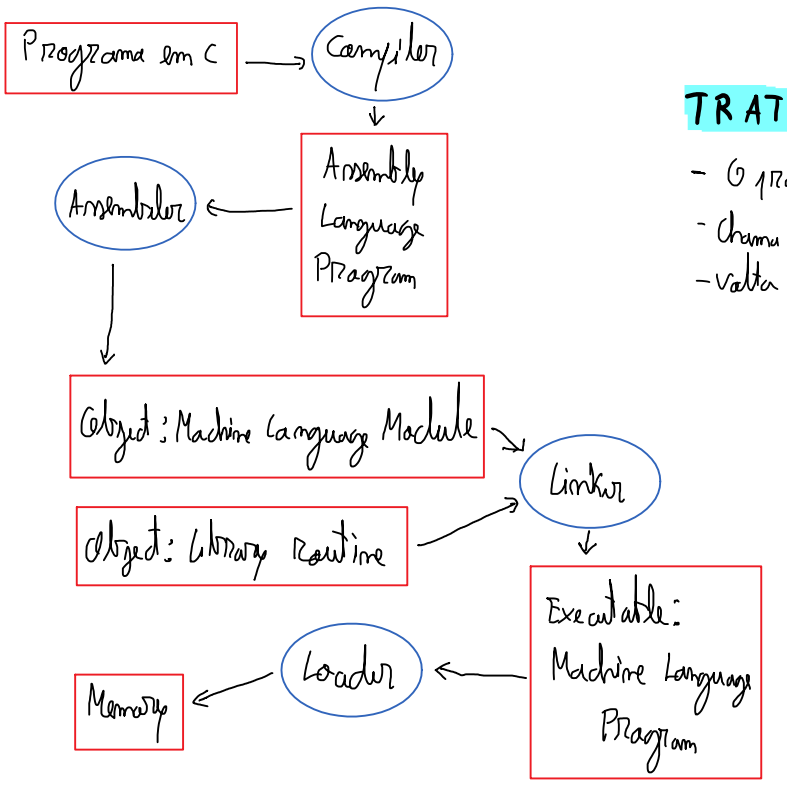
SP - Stack Pointer

1 Salvaguarda de contexto
C> Guarda-se os registos que se vai usar na stack
2 Leitura de operandos da stack
3 Declaração de variáveis
4. Corpo de função
5. Escrita de Resultado na stack
6. Reposição do contexto
7. Retorno da função

Regra geral o SO inicia o SP
Se não houverse SO tem de ser o user a iniciar

Caller : São salvaguardados pela Mãe
Callee : São salvaguardados pela Filha

COMPILAÇÃO



Conversão de base β para dec

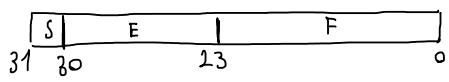
$abcd,efg = a\times\beta^3 + b\times\beta^2 + c\times\beta^1 + d\times\beta^0 + e\times\beta^{-1} + f\times\beta^{-2}$

TIPOS DE DADOS

Char - Byte - 1B = 8 bits
Short - Half Word - 2B = 16 bits
Int/Long - Word - 4B = 32 bits
Long Long - Double Word - 8B = 64 bits
- Quad word - 16B = 128 bits
Single Precision Floating Point - 4B = 32 bits
Double Precision Floating Point - 8B = 64 bits

VIRGULA FLUTUANTE

Norma IEEE-754



$(-1)^S \times M \times B^E$

S - Sinal
E - Expoente (Tem um offset de 127)
F - Fração
M - Mantissa
B - Base

Ex)

$1427 = 0000\ 0101\ 1001\ 0011_2 = 1,0110\ 0101\ 0011 \times 2^{10}$

IEEE-754:

0 | 10001001 | 0110 0101 0011 0000 0000 000

Sinal: 0

Expoente: 127+10 = 137 = 1000 1001

INTERRUPÇÃO & EXCEÇÃO

Evento gerado por uma entidade externa - Mexer no resto

Evento gerado em consequência da execução de uma instrução
- Divisão por 0
- Rotina do SO
- Instrução ilegal

TRATAMENTO DE I/E

- O processador termina a instrução atual
- Chama a subrotina que atende I/E
- volta ao código inicial

Acaba a instrução atual
SEPC ← PC+4
PC ← STVEC

1 Lê a estrutura do executável
2 Carrega as secções de instruções e dados para memória
3 Copia os parâmetros de entrada para a stack
4 Inicializa alguns registos do processador (ex: SP)
5 Salta para início do programa (main)

NOTAS PARA RISC-V

fadd.s -> single (c float)
fadd.d -> double (c double)

Alinhamento em memória
- Byte: Todos os endereços
- HWord: Endereços que acabam em 0 zero
- Word: Endereços que acabam em 2 zeros
- DWord: Endereços que acabam em 3 zeros

PC: Program counter
Ponteiro para a próxima instrução
• Zero N aloca N bytes a 0

! Expoente
01h ... FEh - Caso normal
00h - Formato não normalizado (o offset é 126)
FFh - Se fração = 0 -> +/- ∞
Se fração ≠ 0 -> NaN

PERIFÉRICOS

Port - Mapped I/O
└> Barramentos independentes de acesso à memória e aos periféricos
└> Precisa de instruções específicas

Memory - Mapped I/O (RISC-V)
└> Barramento único para acesso à memória e aos periféricos
└> Usa instruções normais de acesso à memória

SIE (Permite ativar e desativar os I/E)
SIP (Tem as interrupções que ainda falta atender)
SEPC (Guarda a return adress da rotina de I/E)
STVEC (Tem o endereço da rotina que trata de I/E)
SCAUSE (Tem a origem da I/E)
SSTATUS (Tem vários bits com informação do estado do processador tais como:
GIE (Ativa/desativa todos os I/E)
SPIE (Guarda o valor de GIE antes de I/E)

EXTRAS

duet de AQ2,2 e BQ1,3 é CQ 3,5 e tem de se prolongar A e B para 8 bits//

ISA'S

Tipos de máquinas:

- Load / store ou Register - Register machine
  - Acesso à memória é feito por instruções load / store
  - Todos os operandos são registos ou imediatos
  - As instruções têm tipicamente 3 operandos
- Register - Memory machine
  - Multiplos modos de endereçamento:
    - Imediatos
    - Registos
    - Direto
    - Indexado
  - As instruções têm normalmente 2 operandos

RISC: Reduced Instruction Set Computer
- Máquina Store - Load
- Palavra de instrução com tamanho fixo
- Instruções mais simples
- Suporta menos instruções
- Requer mais registos
- Gera programas maiores
- Maior número de instruções por ciclo

CISC: Complex Instruction Set Computer
- Máquina registo - memória
- Palavras de instrução podem ter tamanho variável
- Suporta mais instruções
- Requer menos registos
- Gera programas menores
- Menor número de instruções por ciclo
- A stack tende a ser mais usada

Data - Level Parallelism (DLP)

- SIMD: Single Instruction Multiple Data
  - Registos de dimensão fixa que funcionam como vetores
    - Os membros desses vetores são usados em simultâneo para operações
- SIMT: Single Instruction Multiple Thread
  - Usada em GPU

PROCESSADOR

Instruction Fetch (IF) - PC e memória de instruções

Instruction Decode (ID) - Tem o decoder de instruções

Operand Fetch (OF) - Tem os bancos de registos

Execute (Ex) - Tem as ALU's

Memory Access (MEM) - Tem a memória

Write Back (WB) - Mux's para saber onde escrever

ALU's:
- Ripple - Carry Adder
- Barrel Shifter

2º Teste

16 Hz ⇒ 1MD

PROCESSADOR

Instruction Fetch (IF) - PC e memória de instruções

Instruction Decode (ID) - Tem o decoder de instruções

Operand Fetch (OF) - Tem os bancos de registos

Execute (Ex) - Tem as ALU's

Memory Access (MEM) - Tem a memória e branch control

Write Back (WB) - Mux's para saber onde escrever

ALU's → Ripple-Carry Adder
→ Barrel Shifter

PIPELINING

Ciclo Único

- juntar ID e OF

Desempenho

Million Instruction Per Second (MIPS) = $\frac{\#Inst. / 10^6}{t}$

Floating Point Operations Per Second (FLOPS) = $\frac{\#FPOp}{t}$

Pipeline

- Coloca registos entre os estágios do processador
- No IF, onde se encontra o PC, regista um valor se quando a instrução chega ao MEM
- Em cada ciclo há uma instrução diferente em cada estágio.

(S = s estágios)

	Pipeline vs Single Cycle	
#ciclos para executar N instruções	N + (S − 1)	N
Período de relógio (T clk)	Limitado pelo maior caminho crítico de um estágio	Limitado pelo maior caminho crítico
Tempo de execução	N >> S $T = N \times \hat{T}_{CLK}$	$T = N \times T_{clk}$

$$Speed\ up = \frac{\text{Tempo do original}}{\text{Tempo do melhorado}} = \frac{T}{\hat{T}} = \frac{T_{clk}}{\hat{T}_{clk}}$$

Conflitos:

Regra geral toda a previsão 3 NOP

Estruturais → falta de hardware para realizar uma instrução

Dados → os operandos não estão prontos

Controlo → Originados por qualquer instrução de controlo de fluxo (jumps, conds)

Resolução → Por software → Introduzir NOP's e ter atenção às instruções após um branch
→ Alterar a ordem de instruções

→ Por hardware → Podemos antecipar a estrutura de salto
Colocar no Ex ou até no ID

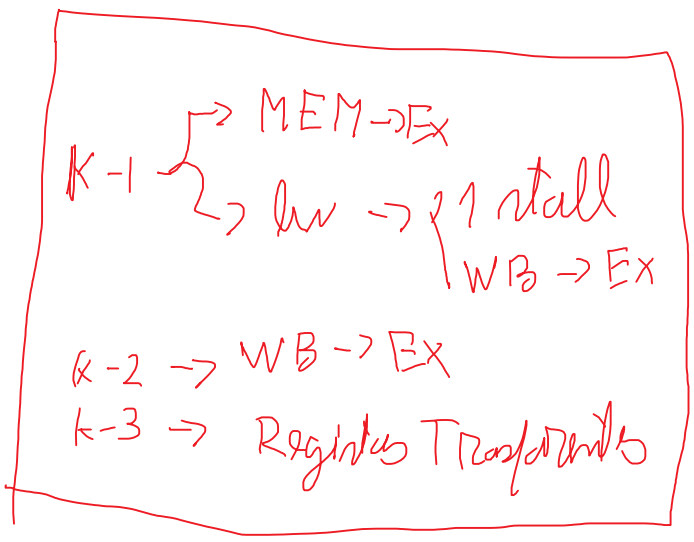
→ Forwarding / Bypassing: (Vão dá para resolver dependências de Loads)

MEM → Ex
WB → Ex
Registos transparentes (WB → ID) → Regra geral aumenta o caminho crítico

→ Execução especulativa: (Invalidate permite fazer previsões erradas)

- Static Branch Prediction: Predict Not Taken (segue sempre em frente)

Exceções podem ser → Recuperáveis (overflow)
→ Não recuperáveis (illegal inst)
→ anulam tudo o que está na pipeline



CACHES:

Localidade temporal:
- Se acedermos a um endereço A, é provável que numa futura próxima necessitemos de voltar a aceder

Localidade espacial:
- Se acedermos a um endereço A, é provável que numa futura próxima necessitemos de aceder a endereços adjacentes

Regra 90/10: Um programa passa 90% do tempo a executar 10% das instruções

- Mapeamento direto -> 1 via
- Cache associativa -> N vias
- Cache completamente associativa -> N vias com 1 linha cada

Bloco = Nº de bits da linha

$offset = \log_2(\text{bloco})$

$index = \log_2\left(\frac{\text{Dim da cache}}{\text{bloco} \times \text{Nº vias}}\right) = \log_2(\text{Nº linhas})$

Tag = Address Size − offset − index

$\text{Hit Rate} = \frac{\#\text{HITS}}{\#\text{Acessos}}$ $\text{Miss Rate} = \frac{\#\text{MISS}}{\#\text{Acessos}}$ $MR = 1 - HR$

$T_{ACESSO} = T_{L1} + MR_{L1} \times T_{RAM} = T_{L1} + MR_{L1} \times (T_{L2} + MR_{L2} \times T_{RAM})$

T_x = Tempo de acesso a x MR_x = Miss Rate de x

Política de substituição

FIFO
- Substitui o bloco que está na cache há mais tempo

- Substitui o bloco que está na cache há mais tempo sem ser acedido

- Substitui ao acaso

- Nota substitui o bloco que foi acedido mais recentemente

Escrita de dados

Memória Virtual

Endereço virtual é composto por:

virtual page nº e offset dentro da página

$offset = \log_2(\text{Tamanho Página})$ Nº de bit da PV = EV − offset

$\text{Nº entradas} = \frac{\text{Nº de bit da PV}}{\text{Nº PTE}}$ $\frac{\text{Tamanho página}}{\text{Tamanho PTE}} = \text{Nº PTE}$

Page Table Root: Regista SATP -> Supervisor Address Translation and Protection

Cada processo tem um SATP

Page Fault: a página não está em RAM

-> existe em disco?

Sim: dá-se uma page swap

Não: dá uma segmentation fault

Uma entrada de uma página (PTE) é composta por:

- P (Present): indica se a página existe
- A (acessed): indica uma página foi usada recentemente
- D (Dirty): indica se a página foi escrita
- R/W (Read/write): indica as permissões de escrita na memória
- Ex (execute): indica as permissões de execução
  - se Ex=1 a página tem instruções
  - se Ex=0 a página tem dados
- U/S (User/Supervisor privilegios): indica as permissões de acesso
- Physical address

TLB (Translation Lookaside Buffer)

-> Uma cache para PTE totalmente associativa

e com bits para o PID (para poder identificar o processo)

Se houver um miss o page table walker vai traduzir e colocar na TLB