

# Lesson 6

# Processor Arch 2

ICS Seminar #9

张龄心

Oct 25, 2023

## 复习: Y86-64指令集

IHALT	0	Code for halt instruction
INOP	1	Code for nop instruction
IRRMOVQ	2	Code for rrmovq instruction
IIRMOVQ	3	Code for irmovq instruction
IRMMOVQ	4	Code for rmmovq instruction
IMRMVQ	5	Code for mrmovq instruction
IOPL	6	Code for integer operation instructions
IJXX	7	Code for jump instructions
ICALL	8	Code for call instruction
IRET	9	Code for ret instruction
IPUSHQ	A	Code for pushq instruction
IPOPQ	B	Code for popq instruction

# SEQ

- 要求:
  - 理解性默写
  - 新指令的填空
- 注意: pushq, popq

Stage	
Fetch	icode:ifun rA:rB valC valP
Decode	$valA \leftarrow R[rA / \%rsp]$ $valB \leftarrow R[rB / \%rsp]$
Execute	$valE \leftarrow (valB / 0) (OP / +) (valA / valC / \pm 8)$ Set CC $Cnd \leftarrow Cond(CC, ifun)$
Memory	$(M_8[valE] \leftarrow valA / valP) / (valM \leftarrow M_8[valE / valA])$
Write back	$R[rB / \%rsp] \leftarrow (if(Cnd)) valE$ $R[rA] \leftarrow valM$
PC update	$PC \leftarrow valP / (Cnd ? valC : valP) / valC / valM$

# SEQ

阶段	pushq rA	popq rA
取指	$\text{icode: ifun} \leftarrow M_1[\text{PC}]$ $\text{rA:rB} \leftarrow M_1[\text{PC}+1]$  $\text{valP} \leftarrow \text{PC}+2$	$\text{icode: ifun} \leftarrow M_1[\text{PC}]$ $\text{rA:rB} \leftarrow M_1[\text{PC}+1]$  $\text{valP} \leftarrow \text{PC}+2$
译码	$\text{valA} \leftarrow R[\text{rA}]$ $\text{valB} \leftarrow R[\%rsp]$	$\text{valA} \leftarrow R[\%rsp]$ $\text{valB} \leftarrow R[\%rsp]$
执行	$\text{valE} \leftarrow \text{valB} + (-8)$	$\text{valE} \leftarrow \text{valB} + 8$
访存	$M_8[\text{valE}] \leftarrow \text{valA}$	$\text{valE} \leftarrow M_8[\text{valA}]$
写回	$R[\%rsp] \leftarrow \text{valE}$	$R[\%rsp] \leftarrow \text{valE}$ $R[\text{rA}] \leftarrow \text{valM}$
更新 PC	$\text{PC} \leftarrow \text{valP}$	$\text{PC} \leftarrow \text{valP}$

# Pipeline

- 流水线5阶段: FDEWM
- 变量名前缀: 大写为该阶段前, 小写为该阶段后 (e\_valE及m\_valM)
- Forward
  - 5个转发源: e\_valE, M\_valE, m\_valM, W\_valE, W\_valM
  - 2个转发目的: d\_valA, d\_valB

```
word d_valA = [  
    D_icode in { ICALL, IJXX } : D_valP;  
    d_srcA == e_dstE : e_valE;  
    d_srcA == M_dstM : m_valM;  
    d_srcA == M_dstE : M_valE;  
    d_srcA == W_dstM : W_valM;  
    d_srcA == W_dstE : W_valE;  
    1 : d_rvalA;  
];
```

# Pipeline

- Pipeline hazard处理: S=stall, B=bubble, N=normal
- stall和bubble的区别
  - Stall: 暂停. 前面所有阶段的指令也要同时stall, 指令不会被丢弃掉.  
为了保证后面的阶段不会空转, 通常要同时在后面插入bubble
  - Bubble: 插入一个nop. 所有指令照常向前移动, 但插入一个bubble, 覆盖接下来本该走到该阶段的指令.  
若要用bubble覆盖某指令, 要确保该指令的所有已进入pipeline的部分都被覆盖掉

条件	流水线寄存器				
	F	D	E	M	W
处理 ret	暂停	气泡	正常	正常	正常
加载/使用冒险	暂停	暂停	气泡	正常	正常
预测错误的分支	正常	气泡	气泡	正常	正常

# Pipeline

- Load/use hazard
  - 条件: E\_icode in {IMRMOVQ, IPOPOPQ} && E\_dstM in {d\_srcA, d\_srcB}
  - 方法: Stall F & D, bubble E

```
word d_valA = [  
    ...  
    d_srcA == M_dstM : m_valM;  
    ...  
];
```

# Pipeline

- Mispredict
  - 条件: `E_icode == IJXX && !e_Cnd`
  - 方法: Bubble D & E
  - 注意PC的更新



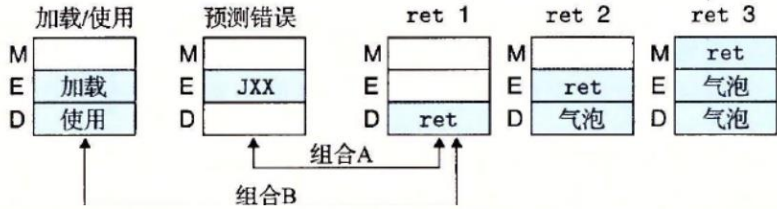
# Pipeline

- Return
  - 条件: IRET in {D\_icode, E\_icode, M\_icode}
  - 方法: Stall F, bubble D
  - 对一条ret指令, 要连续暂停三个周期, 才能彻底清除所有误取的指令

```
word d_valA = [  
    ...  
    d_srcA == M_dstM : m_valM;  
    ...  
];
```

# Pipeline: 多种错误组合

条件	流水线寄存器				
	F	D	E	M	W
处理 ret	暂停	气泡	正常	正常	正常
加载/使用冒险	暂停	暂停	气泡	正常	正常
预测错误的分支	正常	气泡	气泡	正常	正常



条件	流水线寄存器				
	F	D	E	M	W
处理 ret	暂停	气泡	正常	正常	正常
预测错误的分支	正常	气泡	气泡	正常	正常
组合	暂停	气泡	气泡	正常	正常

条件	流水线寄存器				
	F	D	E	M	W
处理 ret	暂停	气泡	正常	正常	正常
加载/使用冒险	暂停	暂停	气泡	正常	正常
组合	暂停	气泡 + 暂停	气泡	正常	正常
期望的情况	暂停	暂停	气泡	正常	正常

# 例

## 6. 判断下列说法的正确性

- (1) (        )流水线的深度越深，总吞吐率越大，因此流水线应当越深越好。
- (2) (        )流水线的吞吐率取决于最慢的流水级，因此流水线的划分应当尽量均匀。
- (3) (        )假设寄存器延迟为 20ps，那么总吞吐率不可能达到或超过 50 GIPS。
- (4) (        )数据冒险总是可以只通过转发来解决。
- (5) (        )数据冒险总是可以只通过暂停流水线来解决。

# 例

6. 判断下列说法的正确性

- (1) (        )流水线的深度越深, 总吞吐率越大, 因此流水线应当越深越好。
- (2) (        )流水线的吞吐率取决于最慢的流水级, 因此流水线的划分应当尽量均匀。
- (3) (        )假设寄存器延迟为 20ps, 那么总吞吐率不可能达到或超过 50 GIPS。
- (4) (        )数据冒险总是可以只通过转发来解决。
- (5) (        )数据冒险总是可以只通过暂停流水线来解决。

**【答】** N, Y, Y, N (mrmov + add), Y

13. 关于流水线技术的描述，错误的是：

- A. 流水线技术能够提高执行指令的吞吐率，但也同时增加单条指令的执行时间。
- B. 减少流水线的级数，能够减少数据冒险发生的几率。
- C. 指令间数据相关引发的数据冒险，都可以通过 data forwarding 来解决。
- D. 现代处理器支持一个时钟内取指、执行多条指令，会增加控制冒险的开销。

# 例

13. 关于流水线技术的描述，错误的是：

- A. 流水线技术能够提高执行指令的吞吐率，但也同时增加单条指令的执行时间。
- B. 减少流水线的级数，能够减少数据冒险发生的几率。
- C. 指令间数据相关引发的数据冒险，都可以通过 data forwarding 来解决。
- D. 现代处理器支持一个时钟内取指、执行多条指令，会增加控制冒险的开销。

答：C，load-use 冒险 不能通过 data forwarding 来解决

## 例

6. 一条三级流水线, 包括延迟为 50ps, 100ps, 100ps 的三个流水级, 每个寄存器的延迟为 10ps。那么这条流水线的总延迟是 \_\_\_\_\_ ps, 吞吐率是 \_\_\_\_\_ GIPS。



## 例

6. 一条三级流水线, 包括延迟为 50ps, 100ps, 100ps 的三个流水级, 每个寄存器的延迟为 10ps。那么这条流水线的总延迟是 \_\_\_\_\_ ps, 吞吐率是 \_\_\_\_\_ GIPS。

【答】 $100+10+100+10+100+10=330\text{ps}$ ,  $1000/(100+10)=9.09$  GIPS



**Thank you!**