Sub-events of Instruction Interpretation

Instruction Fetch
- Generate physical address from PC
- Access cache or memory, if cache miss
- IR ← SR

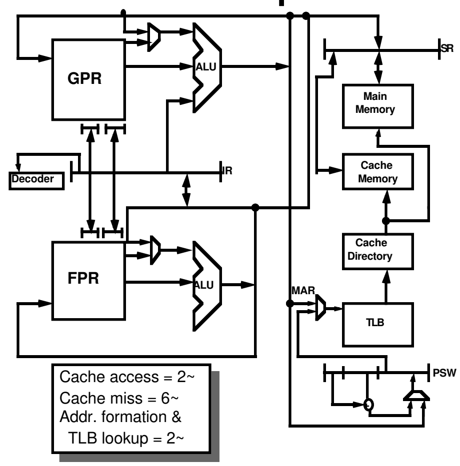
Instruction Decode
- Determine instruction type and addressing modes
- Fetch register operands

Data Fetch
- Generate physical address for operand
- Access cache or memory if cache miss

Execute
- Move SR to ALU, compute result

Update Registers
- Adjust PC to point to next instruction
- Store result in register

The Datapath

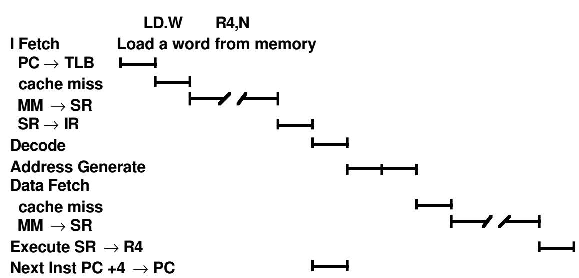


Sample program

"3N+1" Program -- Given an integer, test if it is 1. If so, stop; else test if its odd. If so, triple it and add 1; if not divide by 2 and start again.

```
TNP1     LD.W    R4,N          ; Get operand
         LDA     R5,3[R0,R0]   ; Load 3
Conv:    MOVE    R6,R4         ; Put in temp
         CMP.W   R6,One        ; Converged?
         BC.EQ   Done          ; If yes, quit
EOtest:  AND.W   R6,One        ; Clear all bits but lsb
         BCT     R6,Halve      ; Is number even or odd
Triple:  MPY.H   R4,R5         ; Form product
         INC     R4            ; Bump result
Halve:   SHR.W   R4,1          ; Divide by 2
         BR      Conv          ; Check for convergence
Done:    BR      R15           ; Return to caller
One      WORD    #1d           ; Constant 1
         End
```


Timing for LD.W



Timing for LDA

LDA R5,3[R0,R0]

I Fetch — Load address into register

- PC → TLB
- cache → SR
- SR → IR

Decode

Execute 3+R0+R0 → R5

Next Inst PC +4 → PC

Alternate

LDA R5,3[R0,R0]

I Fetch — Load a word from memory

- PC → TLB
- cache → SR
- SR → IR

Decode

Address Generate

Execute MAR → R5

Next Inst PC +4 → PC

Timing for CMP

CMP.W R6,One

I Fetch — Compare w/ word from memory

- PC → TLB
- cache → SR
- SR → IR

Decode

Address Generate

Data Fetch

- cache → SR

Execute R6 - SR → CC

Next Inst PC +4 → PC

Timing for MPY.H

MPY.H R4,R5

I Fetch — Multiply two registers

- PC → TLB
- cache → SR
- SR → IR

Decode

Execute R4[15:0] * R5[15:0] → R4

Next Inst PC +2 → PC

Timing for BCT

BCT R6,Half

I Fetch — Multiply two registers

- PC → TLB
- cache → SR
- SR → IR

Decode

Execute

- calc branch target addr
- R6 - 1 → R6
- if R6=0
  - then PC +4 → PC
  - else target → PC