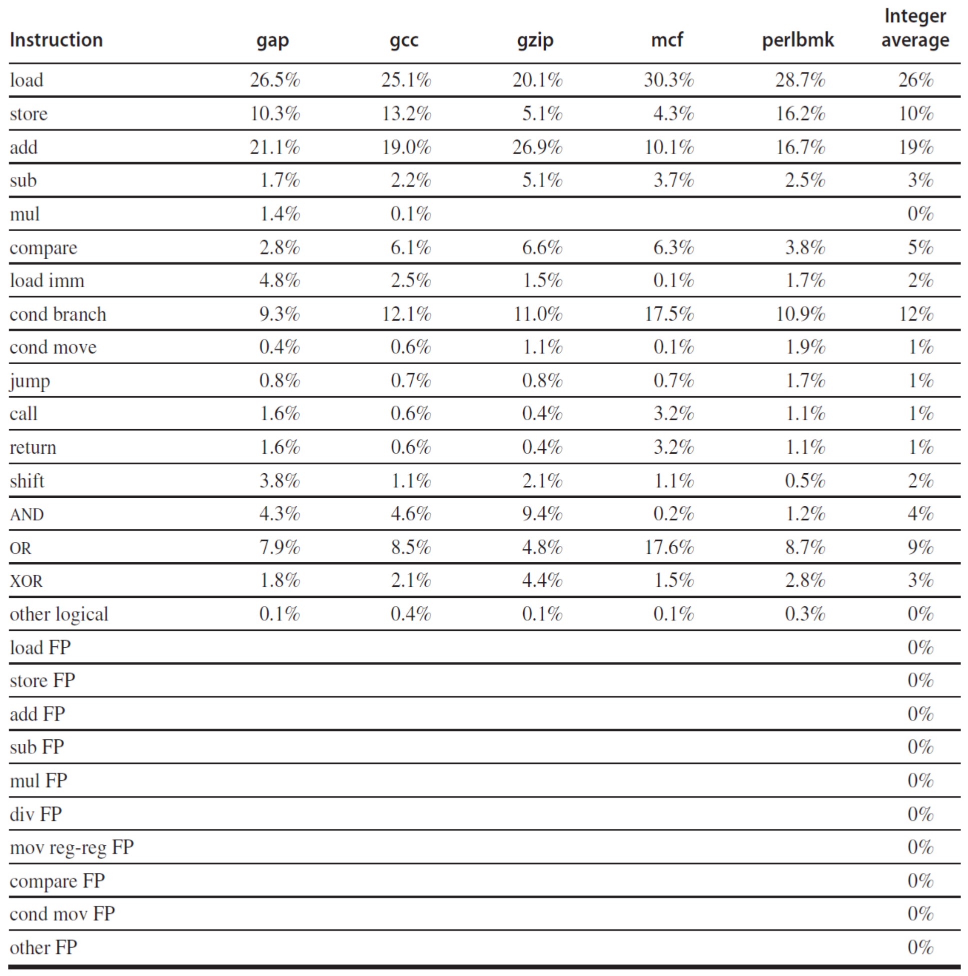
# Recitation 2

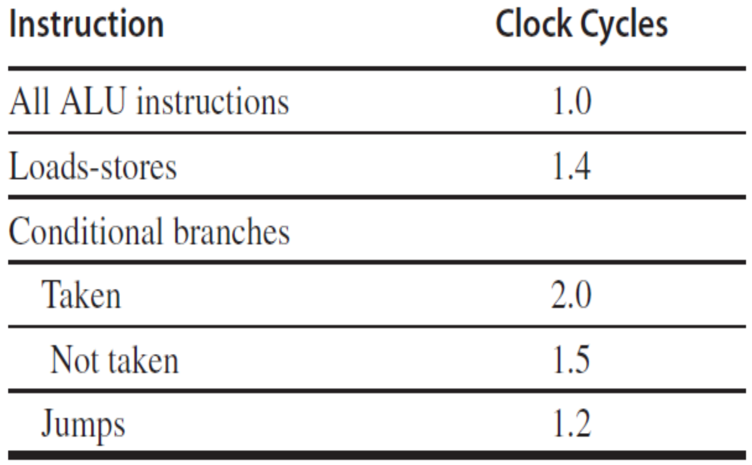
**CS 3853: Computer Architecture**

# CPU Performance



| Instruction | gap | gcc | gzip | mcf | perlbmk | Integer average |
|---|---|---|---|---|---|---|
| load | 26.5% | 25.1% | 20.1% | 30.3% | 28.7% | 26% |
| store | 10.3% | 13.2% | 5.1% | 4.3% | 16.2% | 10% |
| add | 21.1% | 19.0% | 26.9% | 10.1% | 16.7% | 19% |
| sub | 1.7% | 2.2% | 5.1% | 3.7% | 2.5% | 3% |
| mul | 1.4% | 0.1% | | | | 0% |
| compare | 2.8% | 6.1% | 6.6% | 6.3% | 3.8% | 5% |
| load imm | 4.8% | 2.5% | 1.5% | 0.1% | 1.7% | 2% |
| cond branch | 9.3% | 12.1% | 11.0% | 17.5% | 10.9% | 12% |
| cond move | 0.4% | 0.6% | 1.1% | 0.1% | 1.9% | 1% |
| jump | 0.8% | 0.7% | 0.8% | 0.7% | 1.7% | 1% |
| call | 1.6% | 0.6% | 0.4% | 3.2% | 1.1% | 1% |
| return | 1.6% | 0.6% | 0.4% | 3.2% | 1.1% | 1% |
| shift | 3.8% | 1.1% | 2.1% | 1.1% | 0.5% | 2% |
| AND | 4.3% | 4.6% | 9.4% | 0.2% | 1.2% | 4% |
| OR | 7.9% | 8.5% | 4.8% | 17.6% | 8.7% | 9% |
| XOR | 1.8% | 2.1% | 4.4% | 1.5% | 2.8% | 3% |
| other logical | 0.1% | 0.4% | 0.1% | 0.1% | 0.3% | 0% |
| load FP | | | | | | 0% |
| store FP | | | | | | 0% |
| add FP | | | | | | 0% |
| sub FP | | | | | | 0% |
| mul FP | | | | | | 0% |
| div FP | | | | | | 0% |
| mov reg-reg FP | | | | | | 0% |
| compare FP | | | | | | 0% |
| cond mov FP | | | | | | 0% |
| other FP | | | | | | 0% |

**Figure A.27 MIPS dynamic instruction mix for five SPECint2000 programs.** Note that integer register-register move instructions are included in the OR instruction. Blank entries have the value 0.0%.



| Instruction | Clock Cycles |
|---|---|
| All ALU instructions | 1.0 |
| Loads-stores | 1.4 |
| Conditional branches | |
| Taken | 2.0 |
| Not taken | 1.5 |
| Jumps | 1.2 |

1. Compute the effective CPI for MIPS using Figure A.27. Assume we have made the following measurements of average CPI for instruction types in above table:

Assume that 60% of the conditional branches are taken and that all instructions in the "other" category of Figure A.27 are ALU instructions. Average the instruction frequencies of **gap** and **gcc** to obtain the instruction mix.

# CPU Performance

- ALU:

| Instruction | Average of **gap** and **gcc** percentage |
|---|---|
| Add | 20.05 ~ 20.0 |
| Sub | 1.95 ~ 2.0 |
| Mul | 0.8 |
| Compare | 4.4 |
| Load imm | 3.6 |
| Cond move | 0.5 |
| Shift | 2.4 |
| AND | 4.4 |
| OR | 8.2 |
| XOR | 2.0 |
| Other logical | 0.2 |

Total ALU Instruction frequency = 48.5%

Load/Store:

| Instruction | Average of **gap** and **gcc** percentage |
|---|---|
| Load | 25.8 |
| Store | 11.8 |

Total load/store Instruction frequency = 37.6%

Conditional Branches:

| Instruction | Average of **gap** and **gcc** percentage |
|---|---|
| Cond branch | 10.7 |

Total conditional branch Instruction frequency = 10.7%

Jump:

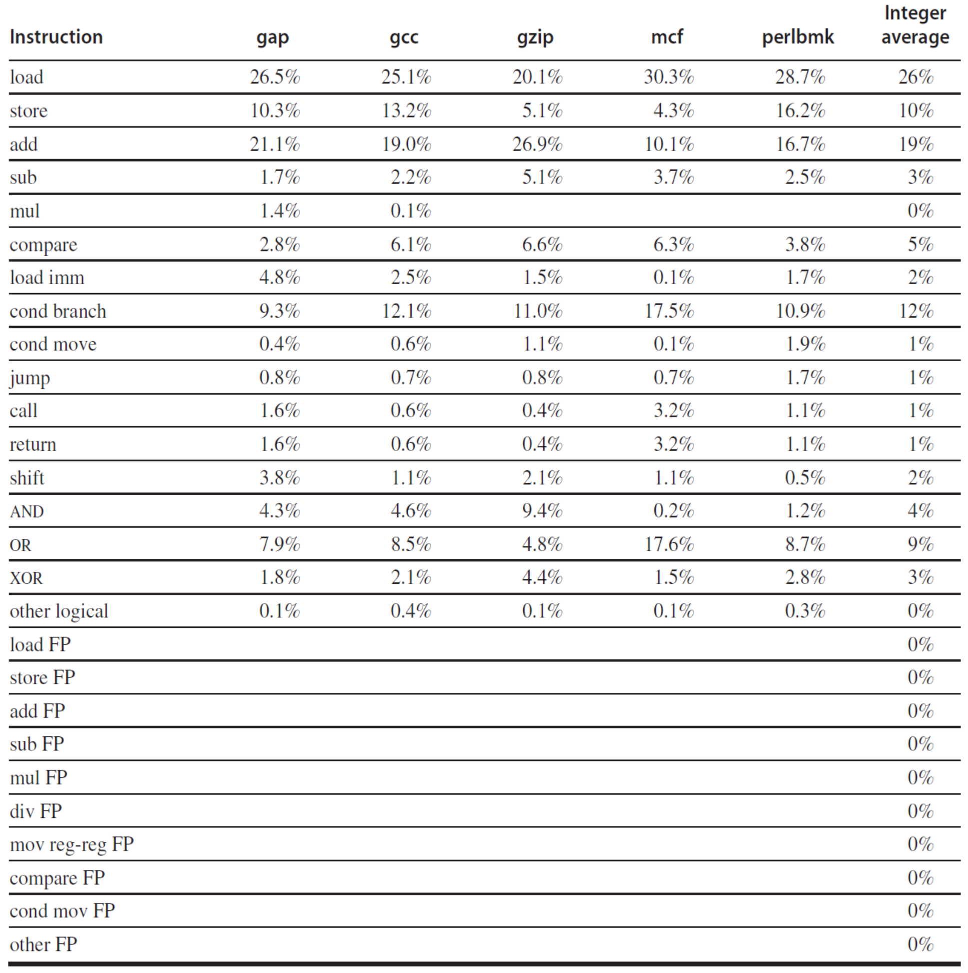
| Instruction | Average of **gap** and **gcc** percentage |
|---|---|
| Jump | 0.8 |
| Call | 1.1 |
| return | 1.1 |

Total jump Instruction frequency = 3%

# CPU Performance

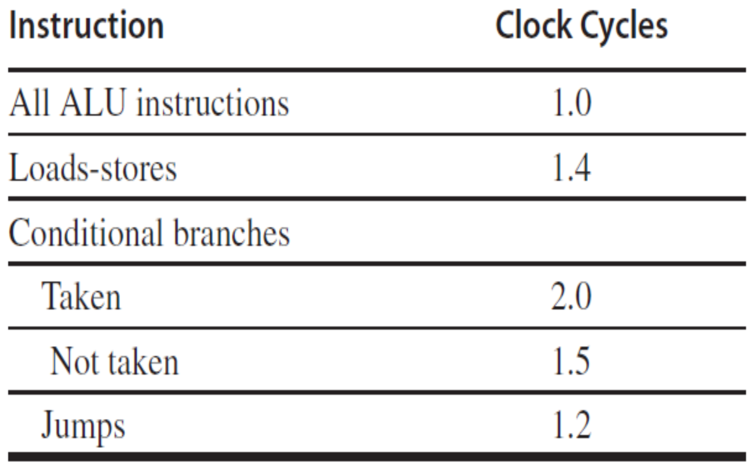
$$
\begin{aligned}
\text{Effective CPI} &= \sum_{\text{categories}} \text{Instruction category frequency} \times \text{Clock cycles for category} \\
&= (\text{ALU instruction frequencies} \times 1.0) + (\text{load/store instruction frequencies} \times 1.4) \\
&\quad + (\text{conditional branch instruction frequencies} \times (0.6 \times 2.0 + (1-0.6) \times 1.5) \\
&\quad + (\text{jump instruction frequencies} \times 1.2) \\
&= (0.485)(1.0) + (0.376)(1.4) + (0.107)((0.6)(2.0) + (1 - 0.6)(1.5)) + (0.03) \times (1.2) \\
&= \mathbf{1.24}
\end{aligned}
$$

# CPU Performance



| Instruction | gap | gcc | gzip | mcf | perlbmk | Integer average |
|---|---|---|---|---|---|---|
| load | 26.5% | 25.1% | 20.1% | 30.3% | 28.7% | 26% |
| store | 10.3% | 13.2% | 5.1% | 4.3% | 16.2% | 10% |
| add | 21.1% | 19.0% | 26.9% | 10.1% | 16.7% | 19% |
| sub | 1.7% | 2.2% | 5.1% | 3.7% | 2.5% | 3% |
| mul | 1.4% | 0.1% | | | | 0% |
| compare | 2.8% | 6.1% | 6.6% | 6.3% | 3.8% | 5% |
| load imm | 4.8% | 2.5% | 1.5% | 0.1% | 1.7% | 2% |
| cond branch | 9.3% | 12.1% | 11.0% | 17.5% | 10.9% | 12% |
| cond move | 0.4% | 0.6% | 1.1% | 0.1% | 1.9% | 1% |
| jump | 0.8% | 0.7% | 0.8% | 0.7% | 1.7% | 1% |
| call | 1.6% | 0.6% | 0.4% | 3.2% | 1.1% | 1% |
| return | 1.6% | 0.6% | 0.4% | 3.2% | 1.1% | 1% |
| shift | 3.8% | 1.1% | 2.1% | 1.1% | 0.5% | 2% |
| AND | 4.3% | 4.6% | 9.4% | 0.2% | 1.2% | 4% |
| OR | 7.9% | 8.5% | 4.8% | 17.6% | 8.7% | 9% |
| XOR | 1.8% | 2.1% | 4.4% | 1.5% | 2.8% | 3% |
| other logical | 0.1% | 0.4% | 0.1% | 0.1% | 0.3% | 0% |
| load FP | | | | | | 0% |
| store FP | | | | | | 0% |
| add FP | | | | | | 0% |
| sub FP | | | | | | 0% |
| mul FP | | | | | | 0% |
| div FP | | | | | | 0% |
| mov reg-reg FP | | | | | | 0% |
| compare FP | | | | | | 0% |
| cond mov FP | | | | | | 0% |
| other FP | | | | | | 0% |

**Figure A.27 MIPS dynamic instruction mix for five SPECint2000 programs.** Note that integer register-register move instructions are included in the OR instruction. Blank entries have the value 0.0%.



| Instruction | Clock Cycles |
|---|---|
| All ALU instructions | 1.0 |
| Loads-stores | 1.4 |
| Conditional branches | |
| Taken | 2.0 |
| Not taken | 1.5 |
| Jumps | 1.2 |

2. Compute the effective CPI for MIPS using Figure A.27 and the table above. Average the instruction frequencies of **gzip** and **perlbmk** to obtain the instruction mix.

# CPU Performance

- ALU:

| Instruction | Average of **gzip** and **perlbmk** percentage |
|---|---|
| Add | 21.8 |
| Sub | 3.8 |
| Mul | 0.0 |
| Compare | 5.2 |
| Load imm | 1.6 |
| Cond move | 1.5 |
| Shift | 1.3 |
| AND | 5.3 |
| OR | 6.8 |
| XOR | 3.6 |
| Other logical | 0.2 |

Total ALU Instruction frequency = 51.1%

Load/Store:

| Instruction | Average of **gzip** and **perlbmk** percentage |
|---|---|
| Load | 24.4 |
| Store | 10.6 |

Total load/store Instruction frequency = 35.0%

Conditional Branches:

| Instruction | Average of **gzip** and **perlbmk** percentage |
|---|---|
| Cond branch | 11.0 |

Total conditional branch Instruction frequency = 11.0%

Jump:

| Instruction | Average of **gzip** and **perlbmk** percentage |
|---|---|
| Jump | 1.2 |
| Call | 0.8 |
| return | 0.8 |

Total jump Instruction frequency = 2.8%

# CPU Performance

$$
\begin{aligned}
\text{Effective CPI} &= \sum_{\text{categories}} \text{Instruction category frequency} \times \text{Clock cycles for category} \\
&= (\text{ALU instruction frequencies} \times 1.0) + (\text{load/store instruction frequencies} \times 1.4) \\
&\quad + (\text{conditional branch instruction frequencies} \times (0.6 \times 2.0+(1-0.6) \times 1.5) \\
&\quad + (\text{jump instruction frequencies} \times 1.2) \\
&= (0.511)(1.0) + (0.35)(1.4) + (0.11)((0.6)(2.0) + (1 - 0.6)(1.5)) + (0.028) \times (1.2) \\
&= \mathbf{1.23}
\end{aligned}
$$

# CPU Performance

| Instruction | applu | art | equake | lucas | swim | FP average |
|---|---|---|---|---|---|---|
| load | 13.8% | 18.1% | 22.3% | 10.6% | 9.1% | 15% |
| store | 2.9% | | 0.8% | 3.4% | 1.3% | 2% |
| add | 30.4% | 30.1% | 17.4% | 11.1% | 24.4% | 23% |
| sub | 2.5% | | 0.1% | 2.1% | 3.8% | 2% |
| mul | 2.3% | | | 1.2% | | 1% |
| compare | | 7.4% | 2.1% | | | 2% |
| load imm | 13.7% | | 1.0% | 1.8% | 9.4% | 5% |
| cond branch | 2.5% | 11.5% | 2.9% | 0.6% | 1.3% | 4% |
| cond mov | | 0.3% | 0.1% | | | 0% |
| jump | | | 0.1% | | | 0% |
| call | | | 0.7% | | | 0% |
| return | | | 0.7% | | | 0% |
| shift | 0.7% | | 0.2% | 1.9% | | 1% |
| AND | | | 0.2% | 1.8% | | 0% |
| OR | 0.8% | 1.1% | 2.3% | 1.0% | 7.2% | 2% |
| XOR | | 3.2% | 0.1% | | | 1% |
| other logical | | | 0.1% | | | 0% |
| load FP | 11.4% | 12.0% | 19.7% | 16.2% | 16.8% | 15% |
| store FP | 4.2% | 4.5% | 2.7% | 18.2% | 5.0% | 7% |
| add FP | 2.3% | 4.5% | 9.8% | 8.2% | 9.0% | 7% |
| sub FP | 2.9% | | 1.3% | 7.6% | 4.7% | 3% |
| mul FP | 8.6% | 4.1% | 12.9% | 9.4% | 6.9% | 8% |
| div FP | 0.3% | 0.6% | 0.5% | | 0.3% | 0% |
| mov reg-reg FP | 0.7% | 0.9% | 1.2% | 1.8% | 0.9% | 1% |
| compare FP | | 0.9% | 0.6% | 0.8% | | 0% |
| cond mov FP | | 0.6% | | 0.8% | | 0% |
| other FP | | | | 1.6% | | 0% |

**Figure A.28 MIPS dynamic instruction mix for five programs from SPECfp2000.** Note that integer register-register move instructions are included in the OR instruction. Blank entries have the value 0.0%.

| Instruction | Clock Cycles |
|---|---|
| All ALU instructions | 1.0 |
| Loads-stores | 1.4 |
| Conditional branches: | |
| Taken | 2.0 |
| Not taken | 1.5 |
| Jumps | 1.2 |
| FP multiply | 6.0 |
| FP add | 4.0 |
| FP divide | 20.0 |
| Load-store FP | 1.5 |
| Other FP | 2.0 |

3. Compute the effective CPI for MIPS using Figure A.28. Assume we have made the following measurements of average CPI for instruction types in table above:

Assume that 60% of the conditional branches are taken and that all instructions in the "other" category of Figure A.28 are ALU instructions. Average the instruction frequencies of **lucas** and **swim** to obtain the instruction mix.

# CPU Performance

- ALU:

| Instruction | Average of **lucas** and **swim** percentage |
|---|---|
| Add | 17.8 |
| Sub | 3.0 |
| Mul | 0.6 |
| Compare | 0.0 |
| Load imm | 5.6 |
| Cond move | 0.0 |
| Shift | 1.0 |
| AND | 0.9 |
| OR | 4.1 |
| XOR | 0.0 |
| Other logical | 0.0 |

Total ALU Instruction frequency = 33%

## Load/Store:

| Instruction | Average of **lucas** and **swim** percentage |
|---|---|
| Load | 9.8 |
| Store | 2.4 |

Total load/store Instruction frequency = 12.2%

## Conditional Branches:

| Instruction | Average of **lucas** and **swim** percentage |
|---|---|
| Cond branch | 1.0 |

Total conditional branch Instruction frequency = 1.0%

## Jump:

| Instruction | Average of **lucas** and **swim** percentage |
|---|---|
| Jump | 0.0 |
| Call | 0.0 |
| return | 0.0 |

Total jump Instruction frequency = 0.0%

## CPU Performance

- FP Load/Store:

| Instruction | Average of **lucas** and **swim** percentage |
|---|---|
| Load FP | 16.5 |
| Store FP | 11.6 |

Total FP load/store Instruction frequency = 28.1%

FP Mul:

| Instruction | Average of **lucas** and **swim** percentage |
|---|---|
| Mul FP | 8.2 |

Total FP mul Instruction frequency = 8.2%

FP Add:

| Instruction | Average of **lucas** and **swim** percentage |
|---|---|
| Add FP | 8.6 |
| Sub FP | 6.2 |

Total FP add Instruction frequency = 14.8%

Other FP:

| Instruction | Average of **lucas** and **swim** percentage |
|---|---|
| mov reg-reg FP | 1.4 |
| compare FP | 0.4 |
| cond mov FP | 0.4 |
| other FP | 0.8 |

Total Other FP Instruction frequency = 3.0%

FP Div:

| Instruction | Average of **lucas** and **swim** percentage |
|---|---|
| Div FP | 0.2 |

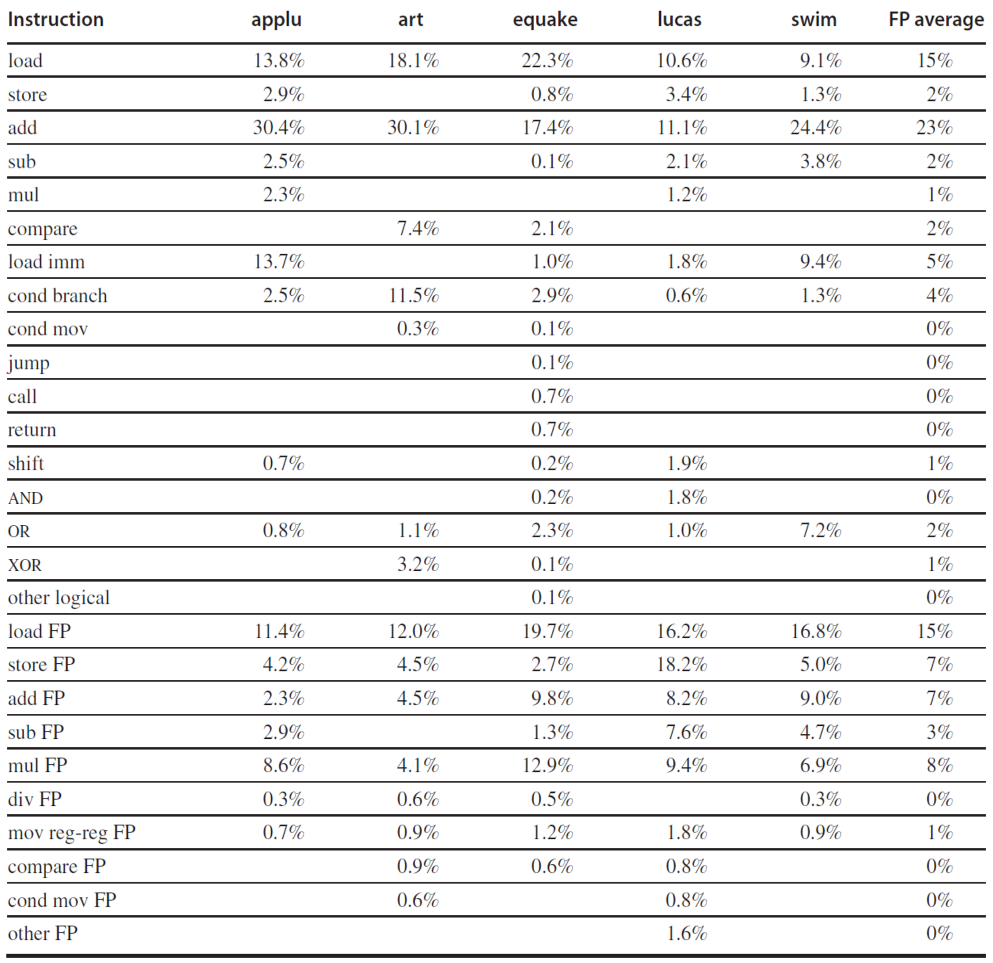
Total FP Div Instruction frequency = 0.2%

# CPU Performance

$$\text{Effective CPI} = \sum_{\text{categories}} \text{Instruction category frequency} \times \text{Clock cycles for category}$$

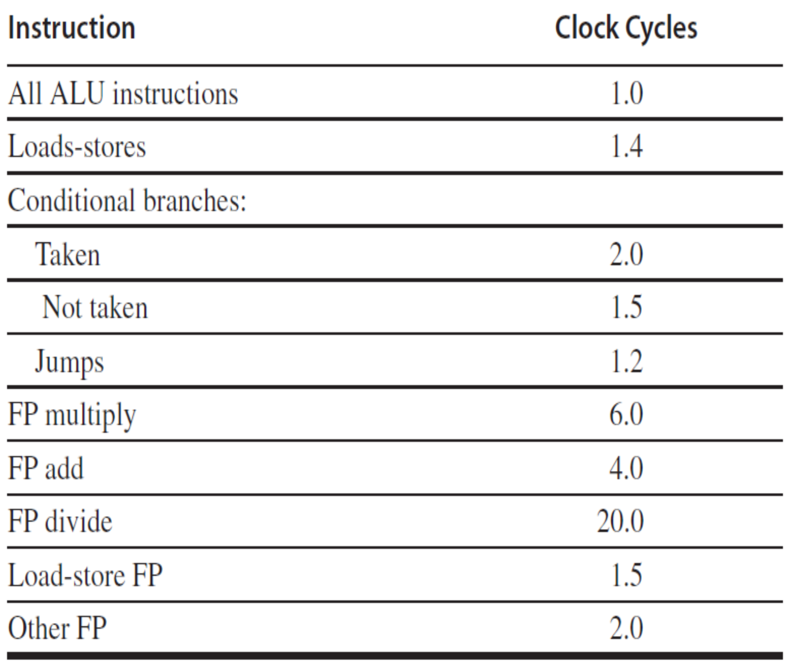
$$\begin{aligned}
&= (\text{ALU instruction frequencies} \times 1.0) + (\text{load/store instruction frequencies} \times 1.4) \\
&\quad + (\text{conditional branch instruction frequencies} \times (0.6 \times 2.0 + (1-0.6) \times 1.5) \\
&\quad + (\text{jump instruction frequencies} \times 1.2) + (\text{FP add instruction frequencies} \times 4.0) \\
&\quad + (\text{FP multiply instruction frequencies} \times 6.0) \\
&\quad + (\text{FP divide instruction frequencies} \times 20.0) \\
&\quad + (\text{Load-store FP instruction frequencies} \times 1.5) \\
&\quad + (\text{Other FP instruction frequencies} \times 2.0) \\
&= (0.33)(1.0) + (0.122)(1.4) + (0.01)((0.6)(2.0) + (1 - 0.6)(1.5)) + (0.0) \times (1.2) \\
&\quad + (0.148) \times (4.0) + (.082) \times (6.0) + (.002) \times (20.0) + (0.281) \times (1.5\ ) + (0.03) \times (2.0) \\
&= \mathbf{2.12}
\end{aligned}$$

# CPU Performance



| Instruction | applu | art | equake | lucas | swim | FP average |
|---|---|---|---|---|---|---|
| load | 13.8% | 18.1% | 22.3% | 10.6% | 9.1% | 15% |
| store | 2.9% | | 0.8% | 3.4% | 1.3% | 2% |
| add | 30.4% | 30.1% | 17.4% | 11.1% | 24.4% | 23% |
| sub | 2.5% | | 0.1% | 2.1% | 3.8% | 2% |
| mul | 2.3% | | | 1.2% | | 1% |
| compare | | 7.4% | 2.1% | | | 2% |
| load imm | 13.7% | | 1.0% | 1.8% | 9.4% | 5% |
| cond branch | 2.5% | 11.5% | 2.9% | 0.6% | 1.3% | 4% |
| cond mov | | 0.3% | 0.1% | | | 0% |
| jump | | | 0.1% | | | 0% |
| call | | | 0.7% | | | 0% |
| return | | | 0.7% | | | 0% |
| shift | 0.7% | | 0.2% | 1.9% | | 1% |
| AND | | | 0.2% | 1.8% | | 0% |
| OR | 0.8% | 1.1% | 2.3% | 1.0% | 7.2% | 2% |
| XOR | | 3.2% | 0.1% | | | 1% |
| other logical | | | 0.1% | | | 0% |
| load FP | 11.4% | 12.0% | 19.7% | 16.2% | 16.8% | 15% |
| store FP | 4.2% | 4.5% | 2.7% | 18.2% | 5.0% | 7% |
| add FP | 2.3% | 4.5% | 9.8% | 8.2% | 9.0% | 7% |
| sub FP | 2.9% | | 1.3% | 7.6% | 4.7% | 3% |
| mul FP | 8.6% | 4.1% | 12.9% | 9.4% | 6.9% | 8% |
| div FP | 0.3% | 0.6% | 0.5% | | 0.3% | 0% |
| mov reg-reg FP | 0.7% | 0.9% | 1.2% | 1.8% | 0.9% | 1% |
| compare FP | | 0.9% | 0.6% | 0.8% | | 0% |
| cond mov FP | | 0.6% | | 0.8% | | 0% |
| other FP | | | | 1.6% | | 0% |

**Figure A.28 MIPS dynamic instruction mix for five programs from SPECfp2000.** Note that integer register-register move instructions are included in the OR instruction. Blank entries have the value 0.0%.



| Instruction | Clock Cycles |
|---|---|
| All ALU instructions | 1.0 |
| Loads-stores | 1.4 |
| Conditional branches: | |
| Taken | 2.0 |
| Not taken | 1.5 |
| Jumps | 1.2 |
| FP multiply | 6.0 |
| FP add | 4.0 |
| FP divide | 20.0 |
| Load-store FP | 1.5 |
| Other FP | 2.0 |

4. Compute the effective CPI for MIPS using Figure A.28 and the table 2 above. Average the instruction frequencies of **applu** and **art** to obtain the instruction mix.

# CPU Performance

- ALU:

| Instruction | Average of **applu** and **art** percentage |
|---|---|
| Add | 30.2 |
| Sub | 1.2 |
| Mul | 1.2 |
| Compare | 3.7 |
| Load imm | 6.8 |
| Cond move | 0.2 |
| Shift | 0.4 |
| AND | 0.0 |
| OR | 1.0 |
| XOR | 1.6 |
| Other logical | 0.0 |

Total ALU Instruction frequency = 46.3%

Load/Store:

| Instruction | Average of **applu** and **art** percentage |
|---|---|
| Load | 16.0 |
| Store | 1.4 |

Total load/store Instruction frequency = 17.4%

Conditional Branches:

| Instruction | Average of **applu** and **art** percentage |
|---|---|
| Cond branch | 7.0 |

Total conditional branch Instruction frequency = 7.0%

Jump:

| Instruction | Average of **applu** and **art** percentage |
|---|---|
| Jump | 0.0 |
| Call | 0.0 |
| return | 0.0 |

Total jump Instruction frequency = 0.0%

## CPU Performance

- FP Load/Store:

| Instruction | Average of **applu** and **art** percentage |
|---|---|
| Load FP | 11.7 |
| Store FP | 4.4 |

Total FP load/store Instruction frequency = 16.1%

FP Mul:

| Instruction | Average of **applu** and **art** percentage |
|---|---|
| Mul FP | 6.4 |

Total FP mul Instruction frequency = 6.4%

FP Add:

| Instruction | Average of **applu** and **art** percentage |
|---|---|
| Add FP | 3.4 |
| Sub FP | 1.4 |

Total FP add Instruction frequency = 4.8%

Other FP:

| Instruction | Average of **applu** and **art** percentage |
|---|---|
| mov reg-reg FP | 0.8 |
| compare FP | 0.4 |
| cond mov FP | 0.3 |
| other FP | 0.0 |

Total Other FP Instruction frequency = 1.5%

FP Div:

| Instruction | Average of **applu** and **art** percentage |
|---|---|
| Div FP | 0.4 |

Total FP Div Instruction frequency = 0.4%

# CPU Performance

$$\text{Effective CPI} = \sum_{\text{categories}} \text{Instruction category frequency} \times \text{Clock cycles for category}$$

$$= (\text{ALU instruction frequencies} \times 1.0) + (\text{load/store instruction frequencies} \times 1.4)$$

$$+ (\text{conditional branch instruction frequencies} \times (0.6 \times 2.0+(1\text{-}0.6) \times 1.5)$$

$$+ (\text{jump instruction frequencies} \times 1.2) + (\text{FP add instruction frequencies} \times 4.0)$$

$$+ (\text{FP multiply instruction frequencies} \times 6.0)$$

$$+ (\text{FP divide instruction frequencies} \times 20.0)$$

$$+ (\text{Load-store FP instruction frequencies} \times 1.5)$$

$$+ (\text{Other FP instruction frequencies} \times 2.0)$$

$$= (0.463)(1.0) + (0.174)(1.4) + (0.07)((0.6)(2.0) + (1 - 0.6)(1.5)) + (0.0) \times (1.2)$$

$$+ (0.048) \times (4.0) + (.064) \times (6.0) + (.004) \times (20.0) + (0.161) \times (1.5\ ) + (0.015) \times (2.0)$$

$$= \mathbf{1.76}$$