April Duff

a = b + c;

b = a + c;

d = a – b;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Accumulator** | **Memory-memory** | **Stack** | **Load-store** |
| Instruction set | load b  add c  store a  load a  add c  store b  load a  sub b  store d | add a, b, c  add b, a, c  sub d, a, b | push b  push c  add  pop a  push a  push c  add  pop b  push a  push b  sub  pop d | load R1, B  load R2, C  add R3, R1, R2  store a, R3  load R1, a  load R2, c  add R3, R1, R2  store b, R3  load R1, a  load R2, b  sub R3, R1, R2  store d, R3 |
| Instructions | 9 | 3 | 12 | 12 |
| Code bytes | 22 | 21 | 27 | 29 |
| Data bytes | 28 | 36 | 28 | 20 |

Which architecture is most efficient as measured by code size?

* Memory-memory

Which architecture is most efficient as measured by total bandwidth required (code + data)?

* Load-store

If the answers are not the same, why are they different?

* The load-store has the best bandwidth because of its registers which can be used as operand sources and result destinations. Although memory-memory is the most efficient according to code size, it does not use registers which can make data retrieval longer especially since it is three at a time in this case.