1.

**SOLUTION:**

Code Sequence:

C=A+B

D=A-E

F=C+D

1. Accumulator Architecture

Load A

Add B

Store C

Load A

Sub E

Store D

Load C

Add D

Store F

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Code | Destroyed Variable | Overhead | Bits(Code Size) | Size of Memory Data |
| Load A |  |  | 40 | 32 |
| Add B | A |  | 40 | 32 |
| Store C |  |  | 40 | 32 |
| Load A | C | 1 | 40 | 32 |
| Sub E | A |  | 40 | 32 |
| Store D |  |  | 40 | 32 |
| Load C |  |  | 40 | 32 |
| Add D |  | 1 | 40 | 32 |
| Store F |  |  | 40 | 32 |
| Total |  | 2x4=8 bytes | 360 bits | 288 bits |

1. Memory-Register Architecture

Load R1,A

Add R2,R1,B

Store R2,C

Sub R3,R1,E

Store R3,D

Add R4,R2,D

Store R4,F

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Code | Destroyed Variable | Overhead | Bits (Code Size) | Size of Memory Data |
| Load R1,A |  |  | 48 | 32 |
| Add R2,R1,B |  |  | 56 | 32 |
| Store R2,C |  |  | 48 | 32 |
| Sub R3,R1,E |  |  | 56 | 32 |
| Store R3,D |  |  | 48 | 32 |
| Add R4,R2,D |  |  | 56 | 32 |
| Store R4,F |  |  | 48 | 32 |
| Total |  |  | 360 bits | 224 bits |

1. Register Architecture

Load R1,A

Load R2,B

Add R3,R1,R2

Store R3,C

Load R4,E

Sub R5,R1,R4

Store R5,D

Add R6,R3,R5

Store R6,F

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Code | Destroyed Variable | Overhead | Bits (Code Size) | Size of Memory Data |
| Load R1,A |  |  | 48 | 32 |
| Load R2,B |  |  | 48 | 32 |
| Add R3,R1,R2 |  |  | 32 |  |
| Store R3,C |  |  | 48 | 32 |
| Load R4,E |  |  | 48 | 32 |
| Sub R5,R1,R4 |  |  | 32 |  |
| Store R5,D |  |  | 48 | 32 |
| Add R6,R3,R5 |  |  | 32 |  |
| Store R6,F |  |  | 45 |  |
| Total |  |  | 384 bits | 192 bits |

2.

**SOLUTION:**

**1.)**

Size of foo struct = 1+1+4+8+2+4+8+8+8+4= 48 bytes

**2.)**

|  |  |
| --- | --- |
| Data Types | Data Size on 64 bit machine(In bytes) |
| Char | 1 |
| Bool | 1 |
| Padding for int | 2 |
| Int | 4 |
| Double | 8 |
| Short | 2 |
| Padding for float | 2 |
| Float | 4 |
| Double | 8 |
| Pointer | 8 |
| Pointer | 8 |
| Int | 4 |
| Trailing padding | 0 |
| Total | 52 |

Required memory size is 52 bytes.

**3.)**

Re arranging size of struct as Highest to lowest we have:

Data Type Data size on 64-bit machine (bytes)

double 8

double 8

pointer 8

pointer 8

int 4

int 4

float 4

short 2

char 1

bool 1

trailing pad 0

Total 48

So, We would need minimum 48 bytes.

3.

**SOLUTION:**

Remaining Instructions make up another 5%

So,

CPI = (0.15 + 0.25) x 4 + 0.15 x 3 + (0.35 + 0.05) x 1 + 0.05 x 12 = 1.6 + 0.45 + 0.4 + 0.6

=3.05

Since Cycles can not be in fraction. So, 3 cycles per instruction.

4.

**SOLUTION:**

Since A B C D are double precision floating points we must use Risc V D

extension.

Firstly load A and B in memory

Fld f1, A; // load A in f1

Fld f2, B; // load B in f2

Fsub f3,f1,f2; // Compute value C and store in f3

//Now for D = 2- A + B

Fadd f7,f1,f2; // storing a+b in reg f7

Fsub f4, 2,f7 // storing 2-A+B in D as f4

//Now we first need to load integer I and J from memory:

Lw x5, 150[I] // load I in X5

Lw x6, 200[J] // load J in X6

// now after having I and j stored in x5 and x6 respectively we can start if else computation.

Bne x5,x6,Else; // If(i==j)

Fadd f1,f1,f2; // A=A+B

Beq x0,x0,Exit; // unconditional exit

Else: Fsub f2,f2,f1; // B=B-A

Exit: // Assembler calculates address.

5.

**SOLUTION:**

**1.)**

Ld x10,d; // load d in x10

Addi x11,x0,0; // i=0 in register x11

Ld x9,0(x0); // X[0] is in & 0(x0) is in reg x9

Lstart:

Ld x12,0(x9) // load X[i] as x12

Add x12,x12,x10; // X[i]+=d;

Addi x9,x9,8; // & x[i++]

Addi x11,x11,1; // i++

Addi x13, x0,100; // x13= 100

Blt x11,x13,Lstart; //loop between current I and I=100

**2.)**

We assumed opcode is 8 bit , memory is 64 bit, and register addresses are 6 bits So,

1) 8+6+6+64= 84 bits

2) 8+6+6+6=26 bits

3) 84 bits

4) 84 bits

5) 8+6+6+6= 26 bits

6) 26 bits

7) 26 bits

8) 26 bits

9) 26 bits

Total code size in bits = 408 bits

= 102 bytes

**3.)**

Total number of load instructions : = 2+100=102

Branches = 100 instructions

ALU: 400 +1 = 401instructions

Now the total number of instructions are 603 instructions

CPI = ((102/603) x 4 )+ ((100/603) x 3) + ((401/603) x 1)

= 0.67 + 0.49 + 0.66

= 1.82

~2

Approximately 2 cycles per instruction