Assignment 1
CMSC611-101
Advanced Computer Architecture, Fall 2009

Sandipan Dey

Question 1: *(20 Points)*

You are trying to figure out whether to build a new fabrication facility for your IBM Power5 chips. It costs \$1 billion to build a new fabrication facility. The benefit of the new fabrication is that you predict that you will be able to sell 3 times as many chips at 2 times the price of the old chips. The old chip had an area of 389 mm2, with a defect rate of 0.3 defects per cm2. The new chip will have an area of 186 mm2, with a defect rate of 0.7 defects per cm2. Assume that the wafer has a diameter of 300 mm and that it costs \$500 to fabricate a wafer in either technology. Assume α =4 and that the wafer yield is 100% (i.e., no defective wafers are used). You were previously selling the chips for 40% more than their cost. Note: for this problem you can just calculate the die cost and ignore the costs of testing and packaging the chips.
A) What is the cost of the old Power5 chip?
B) What is the cost of the new Power5 chip?
C) What was the profit on each old Power5 chip?
D) What is the profit on each new Power5 chip?
E) If you used to sell 500,000 old Power5 chips per month, how long will it take to recoup the costs of the new fabrication facility?

Solution:
Given:

$$Cost_{new-fabrication} = 1 billion\ \$ = 10^9 \$$$

$$Cost_{wafer-fabrication} = Wafer\,Cost = 500\ \$$$

$$(Selling\ \ price/chip)_{new} = 2 \times (Selling\ \ price/chip)_{old}$$

$$(\#Chips\ to\ be\ sold)_{new} = 3 \times (\#Chips\ sold)_{old}$$

$$Area_{old} = 389mm^2 = 389 \times (0.1cm)^2 = 3.89cm^2$$

$$Area_{new} = 186mm^2 = 186 \times (0.1cm)^2 = 1.86cm^2$$

$$(Defect/Area)_{old} = 0.3/cm^2$$

$$(Defect/Area)_{new} = 0.7/cm^2$$

$$Wafer\,diameter = 300mm = 300 \times 0.1cm = 30cm$$

$$Wafer\,yield = 100\% = 1$$

$$\alpha = 4$$

$$(Selling\ \ price/chip)_{old} = 1.4 \times (Cost/chip)_{old}$$

Now, we know,

1. Dies per Wafer = $\dfrac{Wafer\ Area}{Die\ \ Area} - \dfrac{Wafer\ Perimeter}{Diagonal\ length\ of\ \ Die}$ (due to Square peg in round hole)

2. Die yield = Wafer yield $\times\left(1+\dfrac{(Defects/\ Area)\times(Die\ \ Area)}{\alpha}\right)^{-\alpha}$

3. Cost of Die = $\dfrac{Wafer\ Cost}{(Dies\,/\,Wafer)\times(Die\ \ yield)}$

<u>From (1)</u>,

$$(Dies\,/\,Wafer)_{old} = \frac{\pi\left(\dfrac{Wafer\ diameter}{2}\right)^2}{Die\ \ Area_{old}} - \frac{\pi(Wafer\ diameter)}{\sqrt{2\times Die\ \ Area_{old}}} = \frac{3.14159\times\left(\dfrac{30}{2}\right)^2}{3.89} - \frac{3.14159\times 30}{\sqrt{2\times 3.89}}$$

$= 181.71 - 33.79 = 147.92$

$$(Dies\,/\,Wafer)_{new} = \frac{\pi\left(\dfrac{Wafer\ diameter}{2}\right)^2}{Die\ \ Area_{new}} - \frac{\pi(Wafer\ diameter)}{\sqrt{2\times Die\ \ Area_{new}}} = \frac{3.14159\times\left(\dfrac{30}{2}\right)^2}{1.86} - \frac{3.14159\times 30}{\sqrt{2\times 1.86}}$$

$= 380.03 - 48.87 = 331.16$

<u>From (2)</u>,

$$(\text{Die yield})_{\text{old}} = \text{Wafer yield}\times\left(1+\frac{(Defects/\ Area)_{old}\times(Die\ \ Area)_{old}}{\alpha}\right)^{-\alpha} = 1\times\left(1+\frac{0.3\times 3.89}{4}\right)^{-4} = 0.36$$

$$(\text{Die yield})_{\text{new}} = \text{Wafer yield}\times\left(1+\frac{(Defects/\ Area)_{new}\times(Die\ \ Area)_{new}}{\alpha}\right)^{-\alpha} = 1\times\left(1+\frac{0.7\times 1.86}{4}\right)^{-4} = 0.32$$

<u>From (3)</u>,

$$(\text{Cost of Die})_{\text{old}} = \frac{Wafer\ Cost}{(Dies\,/\,Wafer)_{old}\times(Die\ \ yield)_{old}} = \frac{500}{147.92\times 0.36} = 9.39\$$$

$$(\text{Cost of Die})_{\text{new}} = \frac{Wafer\ Cost}{(Dies\,/\,Wafer)_{new}\times(Die\ \ yield)_{new}} = \frac{500}{331.16\times 0.32} = 4.72\$$$

Hence, we have,

A) Cost of old chip = 9.39$
B) Cost of new chip = 4.72$
C) Profit on old chip = 40% (9.39)=3.96$
D) Profit on new chip = Selling price – Cost of chip = $2\times(9.39+3.96)-4.72$ = 21.98$
E) Profit/month = $3\times 5\times 10^5\times 21.98$.
   Hence, # months required to recoup the costs of new fabrication facility

$$= \frac{10^9}{3 \times 5 \times 10^5 \times 21.98} = 30.33$$

Question 2: *(20 Points)*
Imagine that your company is trying to decide between a single-processor system and a dual-processor system. The table below gives the performance on two sets of benchmarks: a memory benchmark and a processor benchmark. You know that your application will spend 40% of its time on memory-centric computations and 60% of its time on processor-centric computations.

Chip	# of cores	Clock frequency (MHz)	Memory performance	Dhrystone performance
Athlon 64 X2 4800+	2	2,400	3,423	20,718
Pentium EE 840	2	2,200	3,228	18,893
Pentium D 820	2	3,000	3,000	15,220
Athlon 64 X2 3800+	2	3,200	2,941	17,129
Pentium 4	1	2,800	2,731	7,621
Athlon 64 3000+	1	1,800	2,953	7,628
Pentium 4 570	1	2,800	3,501	11,210
Processor X	1	3,000	7,000	5,000

A) Calculate the weighted execution time of the benchmarks.
B) How much speedup do you anticipate getting if you move from a Pentium 4 570 to an Athlon 64 X2 4800+ on a CPU-intensive application suite?
C) At what ratio of memory to processor computation would the performance of the Pentium 4 570 be equal to the Pentium D 820?

<u>Solution</u>

A) Weighted performance may be calculated by the following:

$$0.4 \times \text{Memory performance} + 0.6 \times \text{Dhrystone performance}$$

Chip	Weighted Execution Time
Athlon 64 X2 4800+	13800
Pentium EE 840	12627
Pentium D 820	10332
Athlon 64 X2 3800+	11453.8
Pentium 4 1 2,800	5665
Athlon 64 3000+	5758
Pentium 4 570	8126.4
Processor X	5800

B) Speedup = $\frac{CPU\ Performance\ in\ Athlon}{CPU\ Performance\ in\ Pentium} = \frac{20718}{3501} = 5.92$

C) Let's assume the ratio be *1: x.* Then, average performance =

$$x \times 11210 + 1 \times 3501 = x \times 15220 + 1 \times 3000$$

$\Rightarrow x = \frac{3501 - 3000}{15220 - 11210} \approx 0.125$. Hence, the ratio at which their performances are equal is

≈1000:125 =8:5.

Question 3: *(20 Points)*
Your company's internal studies show that single-core system is sufficient for the demand on your processing power. You are exploring, however, whether you could save power by using two cores.
A) Assume your application is 100% parallelizable. By how much could you decrease the frequency and get the same performance?
B) Assume that the voltage may be decreased linearly with the frequency. How much dynamic power would the dual-core system require as compared to the single-core system?
C) Now assume the voltage may not decrease below 75% of the original voltage. This voltage is referred to as the "voltage floor," and any voltage lower than that will lose the state. What percent of parallelization gives you a voltage at the voltage floor?
D) How much dynamic power would the dual-core system require form part (A) compared to the single-core system when taking into account the voltage floor?

Solution
A) $\text{Power}_{\text{dynamic}}$=1/2 × Capacitive Load × Voltage^2 × Frequency switched.

So, if we reduce frequency, we can save power. But, reducing frequency (clock rate) will reduce performance as well.

Consider any application program P, which takes *k* CPU clock cycles. Also,
CPU execution time for a program *P* = $\frac{CPU\ clock\ cycles\ for\ P}{clock\ rate} = \frac{k}{f}$
Also, since the application program *P* is 100% parallelizable, it's should take exactly ½ amount of time to execute in 2-core as in single-core, i.e., from program *P*'s point of view, effective clock rate is doubled in dual-core. Let's assume clock frequency in single core is *f*. Then effective clock rate in dual core for the parallelizable program P is *2f*.

Let's say I have decreased the frequency of the clock to f_{new} in the dual core, hence it's effectively 2 f_{new} for the single core and by condition we have, the same performance for both since and dual core, i.e.,

$$\frac{(Performance)_{Single-Core}}{(Performance)_{Dual-Core}} = \frac{(Execution\ time)_{Dual-Core}}{(Execution\ time)_{Single-Core}} = \frac{k/2f_{new}}{k/f} = 1 \Rightarrow f_{new} = \frac{f}{2}$$

Consequently, I can at most reduce the frequency to half to get the same performance.

B) $\text{Power}_{\text{dynamic}}$=1/2 × Capacitive Load × Voltage^2 × Frequency switched.

Since voltage can be reduced with frequency linearly, we have $f_{new} = \frac{f}{2}$ and

$$\frac{f_{new}}{f} = \frac{V_{new}}{V} = \frac{1}{2} \Rightarrow V_{new} = \frac{V}{2}.$$

Hence, $\frac{(Power_{dynamic})_{Dual-Core}}{(Power_{dynamic})_{Single-Core}} = \frac{V_{new}{}^2 \times f_{new}}{V^2 \times f} = \frac{(V/2)^2 \times (f/2)}{V^2 \times f} = \frac{1}{8}$,

Power is 8 times less than single core.

C) Let's assume that the desired parallelization is x%. Then, we have, any frequency f_{new} in dual core equivalent to clock rate $\left(1+\frac{x}{100}\right)f_{new}$ in single core when thought of in terms of execution time of the program and performance (Note that for *x=100* it exactly becomes twice). By the performance equality constraint (as before) on program *P*, we then have

$$\frac{k \Big/ \left(1+\frac{x}{100}\right)f_{new}}{k/f} = 1 \Rightarrow f_{new} = \frac{f}{1+\frac{x}{100}}$$

Also, since voltage linearly decreases with frequency, we have as before,

$\frac{f_{new}}{f} = \frac{V_{new}}{V} = \frac{1}{1+\frac{x}{100}}$, but due to voltage floor constraint, we have, at voltage floor, the following:

$$\frac{V_{new}}{V} = \frac{1}{1+\frac{x}{100}} = \frac{75}{100} \Rightarrow x = 100\left(\frac{4}{3}-1\right) = \frac{100}{3} = 33.33$$

Hence, the desired parallelization is *33.33%*.

D) Again, as before,

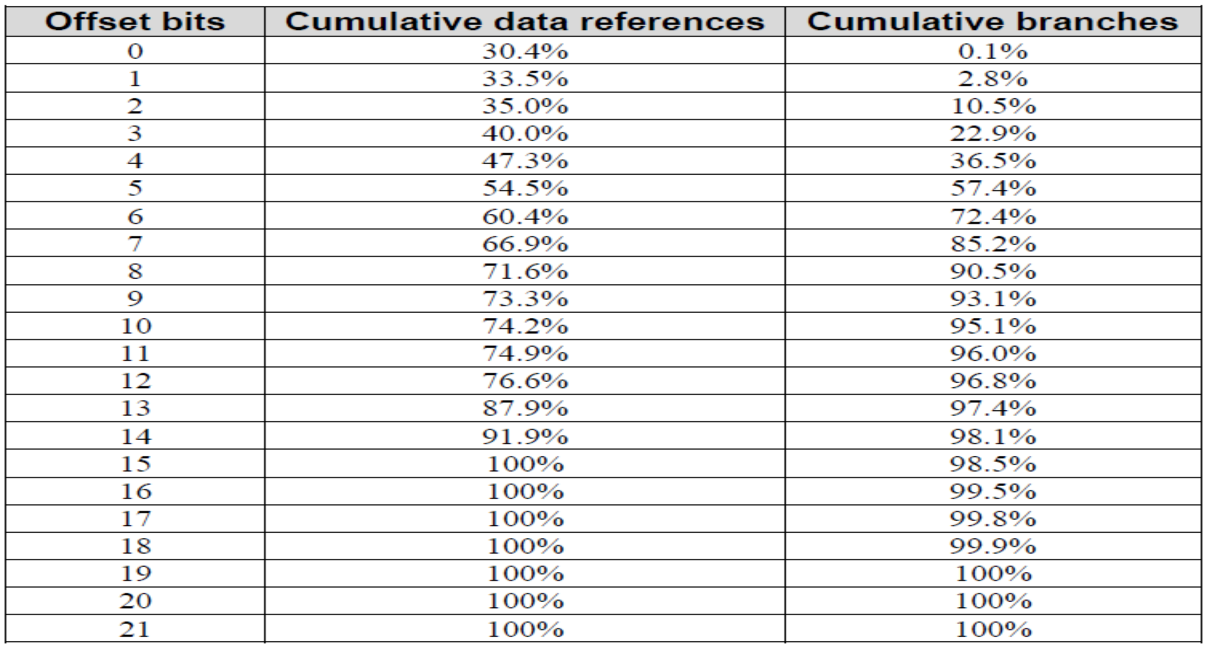
$$\frac{(Power_{dynamic})_{Dual-Core}}{(Power_{dynamic})_{Single-Core}} = \frac{V_{new}{}^2 \times f_{new}}{V^2 \times f} = \frac{\left(1+\frac{x}{100}\right)^{-3} \times V^2 \times f}{V^2 \times f} = \left(1+\frac{x}{100}\right)^{-3} = \left(1+\frac{75}{100}\right)^{-3} \approx 0.19$$

Hence, the power is 19% of the single core.

Question 4: *(20 Points)*

We are designing instruction set formats for a load-store architecture and are trying to decide whether it is worthwhile to have multiple offset lengths for branches and memory references. The length of an instruction would be equal to 16 bits + offset length in bits. ALU instructions will be 16 bits.
Table below contains the data on offset size for the Alpha architecture with full optimization for SPEC CPU2000 in cumulative form. Assume an additional bit is needed for the sign on the offset.
The second and third columns contain the cumulative percentage of data references and branches that can be accommodated with the corresponding number of bits of magnitude in the displacement. For instruction set frequencies, use the data for MIPS from the average of the five benchmarks for the

load-store machine in Figure B.27 on page B-41 (Figure 2.32 on page 138 in 3rd edition) in the textbook. Assume that the miscellaneous instructions are all ALU instructions that use only registers.



Offset bits	Cumulative data references	Cumulative branches
0	30.4%	0.1%
1	33.5%	2.8%
2	35.0%	10.5%
3	40.0%	22.9%
4	47.3%	36.5%
5	54.5%	57.4%
6	60.4%	72.4%
7	66.9%	85.2%
8	71.6%	90.5%
9	73.3%	93.1%
10	74.2%	95.1%
11	74.9%	96.0%
12	76.6%	96.8%
13	87.9%	97.4%
14	91.9%	98.1%
15	100%	98.5%
16	100%	99.5%
17	100%	99.8%
18	100%	99.9%
19	100%	100%
20	100%	100%
21	100%	100%

A) Suppose offsets were permitted to be 0, 8, 16 or 24 bits in length, including the sign bit. What is the average length of an executed instruction?
B) Suppose we wanted a fixed length instruction and we chose a 24-bit instruction length (for everything, including ALU instructions). For every offset of longer than 8 bits, an additional instruction is required. Determine the number of instruction bytes fetched in this machine with fixed instruction size verses those fetched with a byte-variable-sized instruction as defined in part (A).
C) Now suppose we use a fixed offset length of 24 bits so that no additional instruction is ever required. How many instruction bytes would be required? Compare this result to your answer to part (B).

Solution

A) From B-41,
<u>ALU instructions</u> are: add, sub, mul, shift, and, or, xor, other logical and an average (19 + 3 + 0 + 2 + 4 + 9 + 3 + 0) % = 40% time in the benchmarking programs these ALU instructions are used, these instructions will be always 16 bits long, by condition.
<u>Branch instructions</u> are: conditional branch, jump, call, return and an average (12 + 1 + 1 + 1) % = 15% time in the benchmarking programs these branch instructions are used.
There are other <u>data referencing instructions</u> and an average (100 – (40 + 15)) % = 45% time in the benchmarking programs these data referencing instructions are used.

Hence, average length of an executed instruction
$= (16 + 0) \times (1 \times 0.40 + 0.001 \times 0.15 + 0.304 \times 0.45) +$
$(16 + 8) \times (0 \times 0.40 + (0.905 – 0.001) \times 0.15 + (0.716 – 0.304) \times 0.45) +$
$(16 + 16) \times (0 \times 0.40 + (0.995 – 0.905) \times 0.15 + (1 – 0.716) \times 0.45) +$

(16 + 24) × (0 × 0.40 + (1 – 0.995) × 0.15 + (1 - 1) × 0.45)
= 16 × 0.53695 + 24 × 0.321 + 32 × 0.1413 + 48 × 0.00075
= 8.5912 + 7.704 + 4.5216 + 0.03 = 20.8468 bits

B) Let's assume we have *x* instructions in our benchmark program.
With variable size instructions, (average) total program size = 20.8468*x* bits.
With fixed 24 bits instruction size, we have 2 instructions instead of 1 in case of all the instructions that have an offset greater than 8 bits.
Now, in these *x* instructions
number of ALU instructions = 0.40x, they can't have offset.
Only branch and other data referencing instructions can have more than 8 bits offset.
Among 15% branch instructions, there are (100 – 71.6) % = 28.4% of them will have more than 8 bits offset on an average.
Among 45% other data referencing instructions, there are (100 – 90.5) % = 9.5% of them will have more than 8 bits offset on an average.
So, on an average, the number of instructions with more than 8 bits of offset
= (0.40 × 0 + 0.15 × 0.284 + 0.45 × 0.095)*x* = 0.08535*x*.
Hence, total program size = 2 × 24 × 0.08535*x* + 24 × (1- 0.8535)*x* = 26.0484*x*.

$$\frac{\text{number of instruction bytes fetched with fixed instruction size}}{\text{number of instruction bytes fetched with a byte-variable-sized instruction}}$$
$$= \frac{26.0484x}{20.8468x} \approx 1.25$$

C) $$\frac{\#\text{(instruction bytes fetched with fixed instruction size with 24 bits offset w/o additional instruction)}}{\#\text{ (instruction bytes fetched with a byte-variable-sized instruction)}}$$
$$= \frac{(16+24)x}{26.0484x} \approx \frac{40}{26.0484} \approx 1.54$$

Question 5: *(20 Points)*
Compare zero-, one-, two-, and three-address machines by writing programs to compute:
G= (A + B × C) / (D – E × F)
for each of the following four machines. The instructions available for use are as follows:

0 Address	1 Address	2 Address	3 Address
PUSH M POP M ADD SUB MUL DIV	LOAD M STORE M ADD M SUB M MUL M DIV M	MOVE (X ← Y) ADD (X ← X + Y) SUB (X ← X - Y) MUL (X ← X × Y) DIV (X ← X / Y)	MOVE (X ← Y) ADD (X ← Y + Z) SUB (X ← Y - Z) MUL (X ← Y × Z) DIV (X ← Y / Z)

Assume that M is a 16-bit memory address and that X, Y, and Z are either 16-bit memory or 4-bit register addresses. The one address machine uses an accumulator, and the two- and three- address machines have 16-registers and instruction operands can be any combinations of memory locations and registers. Assuming 8-bit opcodes and instruction length that are multiples of 4 bits, how many bits does each machine need to compute G?

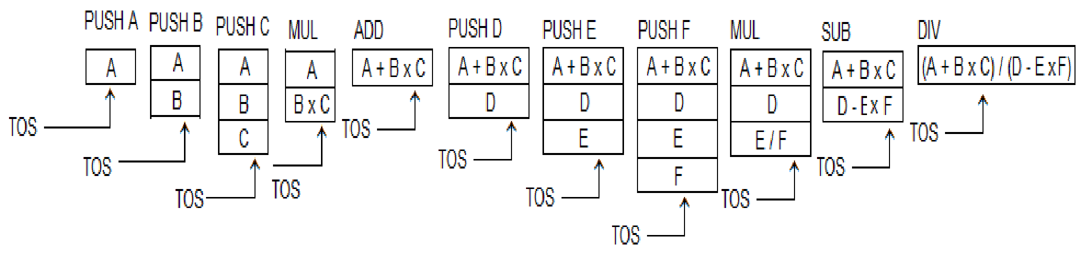
Solution

Here we assume all the operands A, B, C, D, E, F are in the memory.

G = (A + B × C) / (D – E × F)

<u>Postfix</u> form of G = A B C × + D E F × **-** /

0 Address	**Length**	
PUSH A	8 + 16 = 24	TOS←TOS + 1; Stack[TOS]←Memory[A]
PUSH B	8 + 16 = 24	TOS←TOS + 1; Stack[TOS]←Memory[B]
PUSH C	8 + 16 = 24	TOS←TOS + 1; Stack[TOS]←Memory[C]
MUL	8	Stack[TOS-1] ←Stack[TOS-1] × Stack[TOS]; TOS←TOS – 1
ADD	8	Stack[TOS-1] ←Stack[TOS-1] + Stack[TOS]; TOS←TOS – 1
PUSH D	8 + 16 = 24	TOS←TOS + 1; Stack[TOS]←Memory[D]
PUSH E	8 + 16 = 24	TOS←TOS + 1; Stack[TOS]←Memory[E]
PUSH F	8 + 16 = 24	TOS←TOS + 1; Stack[TOS]←Memory[F]
MUL	8	Stack[TOS-1] ←Stack[TOS-1] × Stack[TOS]; TOS←TOS – 1
SUB	8	Stack[TOS-1] ←Stack[TOS-1] - Stack[TOS]; TOS←TOS – 1
DIV	8	Stack[TOS-1] ←Stack[TOS-1] / Stack[TOS]; TOS←TOS – 1
Total	184 bits	



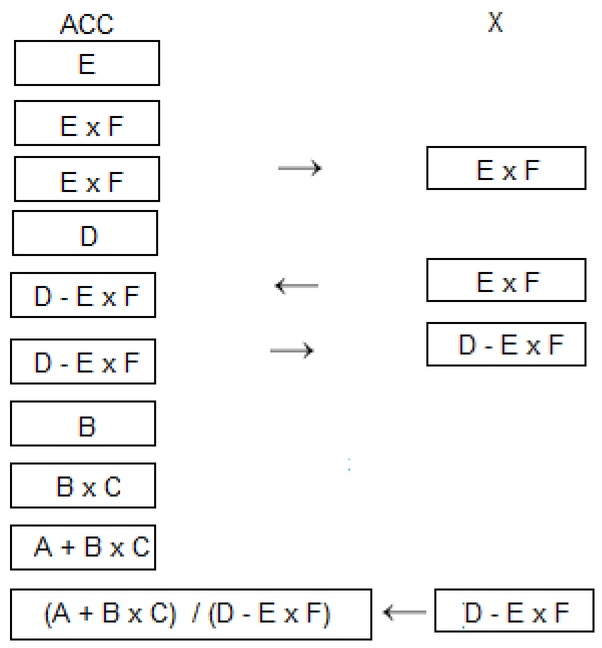
Content of the Stack

According to the priority of the operations, the steps can be:
X = E × F
X = D – X
Y = A + B × C
Z = Y / X

1 Address	**Length**	
LOAD E	8 + 16 = 24	ACC ← E
MUL F	8 + 16 = 24	ACC ← ACC × F
STORE X	8 + 16 = 24	[X] ← [ACC]
LOAD D	8 + 16 = 24	ACC ← D
SUB X	8 + 16 = 24	ACC ← ACC – [X]
STORE X	8 + 16 = 24	[X] ← [ACC]
LOAD B	8 + 16 = 24	ACC ← B

MUL C
ADD A
DIV X
	8 + 16 = 24
	8 + 16 = 24
	8 + 16 = 24
		ACC ← ACC × C
		ACC ← ACC + A
		ACC ← ACC / [X]

Total	240 bits	



2 Address	Length	

MUL B, C
ADD A, B
MUL E, F
SUB D, E
DIV A, D
	8 + 16 + 16 = 40
	8 + 16 + 16 = 40
	8 + 16 + 16 = 40
	8 + 16 + 16 = 40
	8 + 16 + 16 = 40
		B ← B × C
		A ← A+ B
		E ← E × F
		D ← D - E
		A ← A / D

Total	200 bits	

3 Address	Length	

MUL B, B, C
ADD A, A, B
MUL E, E, F
SUB D, D, E
DIV A, A, D
	8 + 16 + 16 + 16 = 56
	8 + 16 + 16 + 16 = 56
	8 + 16 + 16 + 16 = 56
	8 + 16 + 16 + 16 = 56
	8 + 16 + 16 + 16 = 56
		B ← B × C
		A ← A+ B
		E ← E × F
		D ← D - E
		A ← A / D

	280 bits	

Alternatively, if the machine has registers R, S, T, X, Y, Z and we first load all the operands from memory to these registers,

3 Address	Length	

MOV R, A
MOV S, B
MOV T, C
MOV X, D
	8 + 4 + 16 = 28
	8 + 4 + 16 = 28
	8 + 4 + 16 = 28
	8 + 4 + 16 = 28
		R ← A
		S ← B
		T ← C
		X ← D

MOV Y, E MOV Z, F MUL S, S, T ADD R, R, S MUL Y, Y, Z SUB X, X, Y DIV R, R, X	8 + 4 + 16 = 28 8 + 4 + 16 = 28 8 + 4 + 4 + 4 = 20 8 + 4 + 4 + 4 = 20 8 + 4 + 4 + 4 = 20 8 + 4 + 4 + 4 = 20 8 + 4 + 4 + 4 = 20	$Y \leftarrow E$ $Z \leftarrow F$ $S \leftarrow S \times T$ $R \leftarrow R + S$ $Y \leftarrow Y \times Z$ $X \leftarrow X - Y$ $R \leftarrow R / X$
	268 bits	