$$\frac{(17+t)}{\frac{17}{2}+\frac{t}{1024}} = 77 \qquad 17+t = 77\cdot\frac{17}{2} + \frac{77}{1024}\cdot t \qquad t-\frac{77}{1024}t = 637.5 \Rightarrow t = 689.3 \Rightarrow 690$$

1. [20 marks] Digital Logic.

a) Let 'X' be the ternary connective such that 'Xpqr' is equivalent to '(p /\ q) + (q \/ r)'. '+' is exclusive or. 'F' and 'T' denote the 0-ary connectives 'false' and 'true', respectively.

Using {'X', 'T'}, synthesize: ~p |= =| X p p T

Using {'X', 'F'}, synthesize: p \/ q |= =| X F p q ✓

when 0,1 element is true

b) 'M' is the ternary 'minority' connective. 'Mpqr' is true iff a minority of its arguments is true. 'F' denotes the 0-ary connective 'false'.

p=1

~p = M p F F

Using {'M', 'F'}, synthesize: ~p |= =| M p p F ✓

p=0 M(0,0,0)=1

p=1 M(1,1,0)=0

Using {'M', '~', 'F'}, synthesize: p \/ q |= =| M ~p ~q F ✓

2. [20 marks] Amdahl's Law.

a) On a uniprocessor, moderately serial portion A of program P consumes 27 s, while perfectly parallel portion B consumes 143 s. On a parallel computer, portion A speeds up 2x, while portion B speeds up by the number of processors. Given 1,024 processors, what is the speedup on program P?

$(27+143) \div \left(\frac{27}{2} + \frac{143}{1024}\right) = 12.46$

su = 12.46 times

b) On a uniprocessor, moderately serial portion A of program P consumes 17 s, while perfectly parallel portion B consumes 't' s. On a parallel computer, portion A speeds up 2x, while portion B speeds up by the number of processors. How big must 't' be so that using 1,024 processors gives a speedup of at least 77 times? Round any nonintegral 't' to the next highest integer.

(17+t)/

t = 690 s ✓

5. [20 marks] Pipeline Boxes and Pipeline Latches

```
+-------------------------------------------------------------------+ P
|                                                                   | r
|      <I-cache>              <Register file>           <D-cache>   | o
|                                                                   | c
|                           <Control circuitry>                     | e
|                 +-+             +-+          +-+          +-+     | s
|   <f-box>       | |  <d-box>    | |  <x-box> | |  <m-box> | |  <w-box> s
|                 +-+             +-+          +-+          +-+     | o
|                 f/d             d/x          x/m          m/w     | r
+-------------------------------------------------------------------+
```

Memrefs have 2 m-boxes, and floating-point multiplies have 4 x-boxes. Integer add is denoted xi. Consider the following space-time diagram:

f0 write back 1st time — second time

		1	2	3	4	5	6	7	8	9	0	1	2	3
l.d	f0,0(r1)	f	d	xi	m1	m2	w							
l.d	f2,0(r2)		f	d	xi	m1	m2	w						
mul.d	f4,f0,f2			f	d	s	s	x1	x2	x3	x4	n	w	
s.d	f4,0(r1)				f	s	s	d	xi	s	s	m1	m2	

(before compressing

But the actual physical behavior of the pipeline is better represented by the following space-time diagram:

a). source & target. f4 created ~~f4~~ in 10, ~~to~~ then X/m latch. f4 → m1.
data movement answer. f4 → X/m latch → m1 box

f0/f0 1st time

```
1.d     f0,0(r1)   1 | 2 | 3  | 4  | 5  | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3
                   f | d | xi | m1 | m2 | w |   |   |   |   |   |   |
                  -----------------------------------------------------
1.d     f2,0(r2)     | f | d  | xi | m1 | m2| w |   |   |   |   |   |
                  -----------------------------------------------------
mul.d f4,f0,f2       |   |    |    | f  | d | x1| x2| x3| x4| n | w |
                  -----------------------------------------------------
s.d     f4,0(r1)     |   |    |    |    |   | f | d | xi| m1| m2|
```

fo f2 2nd time f0. (after compressing

b) leave and back in 6, ~~leav~~ at cycle 12, ~~leave~~ f0 push into w, w box doesn't act on it. m/w latch contain f0.

In every cycle, the control circuitry has information about the execution of the program so far, and guides boxes to do the right thing.

a) Decribe all that happens to value 'f4' during cycle 11. Mention relevant boxes and latches. in cycle 10.

~~mul.d~~ f4 was calculated by x4, the multiplication value was put into X/m latch, ← mul.d f4,f0,f2

X and f4 = f0×f2. Then, in "s.d f4,0(r1)". m1-box put value of "f4" into memory address "r1+0".

b) When does the value 'f0' leave the pipeline for the second time? State the cycle in which this happens and the mechanism that flushes 'f0'.

X 'f0' leave at cycle ~~11~~ 11, in the first instruction, 'f0' was written into register, this is the first time, The second time, when instruction 3 was completed. 'f0' leave ~~pipeli~~ pipeline after x-box completes.

c) Describe all the actions of the d-box in cycle 6.

X In cycle 6, ~~the value of at memory f0 was written back~~ the value at memory address "r1+0" was written into register "f0" by w-box. m2-box get the value at memory address "r2+0", and put this value into m/w latch. Instruction 3 ~~does~~, stall, waiting for the value of f0, f2, ~~and~~ Instruction 4 also stalls.

d) Why is it O.K. to overlap the m-box with itself in cycle 5?

X In cycle 5, the m1 and m2 boxes operate on different memory addresses and also different registers, so they can overlap.

Hex table:

0	0000	4	0100	8	1000	c	1100 12
1	0001	5	0101	9	1001	d	1101 13
2	0010	6	0110	a	1010 10	e	1110 14
3	0011	7	0111	b	1011 11	f	1111 15

Hex flips:

0 - f	4 - b
1 - e	5 - a
2 - d	6 - 9
3 - c	7 - 8

Hex powers:

1, 16, 256, 4096

Hex naturals:

a	b	c	d	e	f
10	11	12	13	14	15

8 c) d box in cycle. ~~get f0, f2 operands, f0. d box localize f0,~~
f2: m2→x1. f0: d→d/x latch. w→d.

d) m1, 2 m1 at the same time is illegal, m1 & m2 overlaps is ok. It is pipeline.

4. [20 marks] Instructions with Base Register and Offset.
☆ computers do shift, not arithmetic calculation

Each of a family of computers has 128-bit registers, 64-bit memory addresses and integer adders, and 12-bit immediates. All answers to Qn. 4 must be in 64-bit hexadecimal. If a hex number begins with a single hexit 'x' repeated 'y' times, you may write 'x*' instead of '[x:y]'. Show work---in every part, show the **BINARY** manipulations and/or reasoning that led to the answer.

a) Machine A has 32-bit instructions. Consider 'l.d f8,<immediate>(r5)'. In hex, the 12-bit immediate is '8ab'. What is the 64-bit addend added to the base register 'r5'? 1000, 1010, 1011 ☆ negative number.
0000

13 times

ans: 0000000 000000 8 a b = (0*)8ab X

(F*)8ab
= signed bit

f!! immediate is signed number. negative

b) Machine B has 32-bit instructions. Consider 'bne r3,r5,<immediate>'. In hex, the 12-bit immediate is '7cd'. What is the 64-bit addend added to the base register 'PC'? 1997 × 4 = 7988 3892

(0*)1F34 ✓

7988 = 1111 0011 0100 ☆ put ~~two~~ two 0 at the end left shift. ~~at~~ 0111|1100,1101|00

ans: ~~0 0 1 F 3 4~~

2|7988
2|3994 0
2|1997 0
2|998 1
2|499 0
2|249 1
2|124 1
2|62 0
2|31 0
2|15 1
2|7 1
2|3 1
2|1 1

7 = 4+2+1   l.d. do not need arithmetic   -5

c) Machine C has 64-bit instructions. Consider 'l.d f8,<immediate>(r5)'. In hex, the 12-bit immediate is '7cd'. What is the 64-bit addend added to the base register 'r5'? 1997 7cd is positive

(0*)7cd

ans: ~~0 0 0 7 c d~~

d) Machine D has 64-bit instructions. Consider 'bne r3,r5,<immediate>'. In hex, the 12-bit immediate is '8ab'. What is the 64-bit addend added to the base register 'PC'? 2219 × 8 = 17752

17752 = 1 1 0 0 0 1 0 1 0 1 0 1 1 0 0 0   left shif 3 bits

8 a b 3 zeros bits

$\boxed{(0*)4558}$

ans: ~~0 0 4 5 5 8~~

(F*)C558

5. [20 marks] Fractional-Number Formats.

a) [math] Display the infinite binary expansion of 7 1/20. Fact: 20 = 5 * 4. Use the overbar to indicate repetition of a bitstring. Show work.

$7 \to 111.00\overline{0011}$

ans: $111.00\overline{0011}$

b) [math] Normalize the infinite binary expansion of 7 1/20, adding a scale factor, and preparing to move to a floating-point format. Show work.

$1.1100\overline{0011} \times 2^2$

ans: $1.1100\overline{0011} \times 2^2$

PFP is a slight variant of IEEE floating point. In particular, i) there is no sign bit, ii) the exponent is 8-bit two's complement, iii) the fractional field is 40 bits, and iv) exponents are unbiased and no exponents have been removed to encode special values.   48÷4=12   40÷4=10.

c) Show the register contents of 7 1/20 in PFP in hexadecimal. You may use the repetion convention of Qn. 4. Show work.

$\underbrace{00000010}_{8}$ 1100 0011 0011

ans: 0 2 c 3 3 3 3 3 3 3 3 3

9 times = 02c(3*)

d) Here is the register contents of a PFP number in hexadecimal: '06f0*'. Square this number and express the answer in the same way. Hint: Consider scaling (floating). Show work.   13 = 0d

wow! $1.11100000\ 1000\ 0000 \times 2^{13}$

$1.11100\cdots0 = (1.9375 \times 2^6)^2 = 124$

0 000 0110 ⇒ 6 $124^2 = 15376$ ⇒ 11 1100 0000 | 0000

ans: 0 d e 0 8 0 000000

= 0de08(0*)

e) What is the difference between the largest PFP number and the next largest PFP number? Express your answer using powers of 2. Show work.

exponent · $01111111 = 2^7 - 1 = 127$ 1.1111....1

$2^{127} \times 2^{-40} = 2^{87}$

ans: 2^{87}

2 | 17752	
2 | 8876	0
2 | 4438	0
2 | 2219	0
2 | 1109	1
2 | 554	1
2 | 277	0
2 | 138	1
2 | 69	0
2 | 34	1
2 | 17	0

8

2 | 8	1
2 | 4	0
2 | 2	0
2 | 1	0
0	1

$\frac{1}{20} \times 2 = \frac{2}{20}$ 0

$\frac{2}{20} \times 2 = \frac{4}{20}$ 0

$\frac{4}{20} \times 2 = \frac{8}{20}$ 0

$\frac{8}{20} \times 2 = \frac{16}{20}$ 0

$\frac{16}{20} \times 2 = \frac{12}{20}$ 1

$\frac{12}{20} \times 2 = \frac{4}{20}$ 1

$\frac{4}{20} \times 2 = \frac{8}{20}$ 0

$\frac{8}{20} \times 2 = \frac{16}{20}$ 0

$\frac{16}{20} \times 2 = \frac{12}{20}$ 1

$\frac{12}{20} \times 2 = \frac{4}{20}$ 1

$\frac{4}{20} \times 2 = \frac{8}{20}$ 0

$11 = 1.1 \times 2^1$

3 12 1 0

$16^3\ 16^2\ 16^1\ 16^0$

3088

16