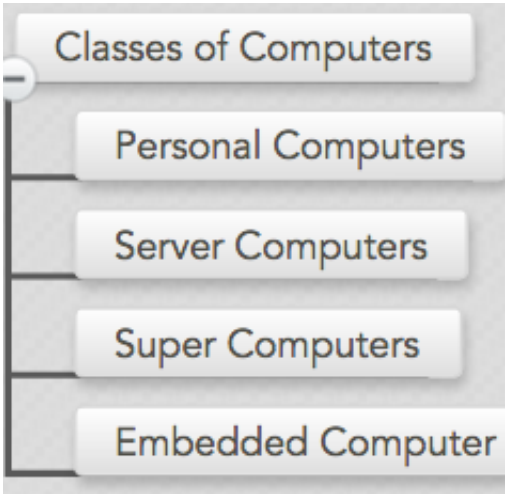
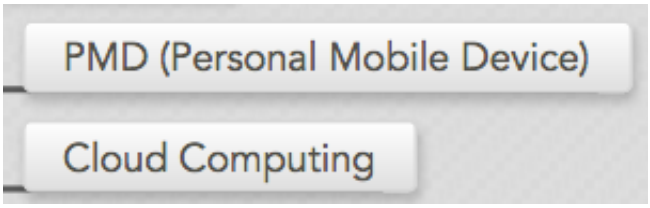
CE450 Q&As

1. Theory Questions

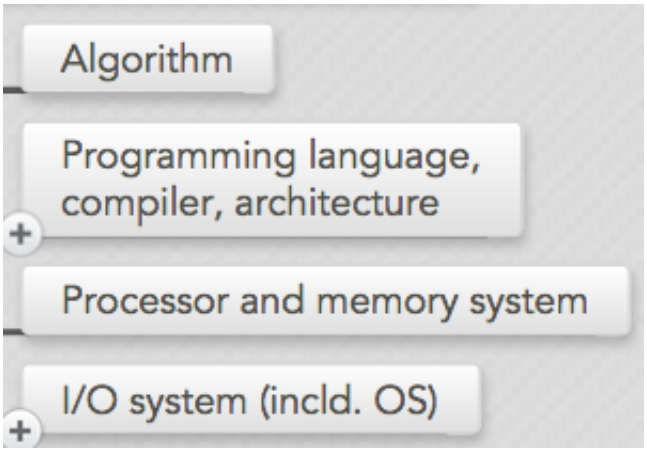
1.1. What are the 4 Classes of Computers?



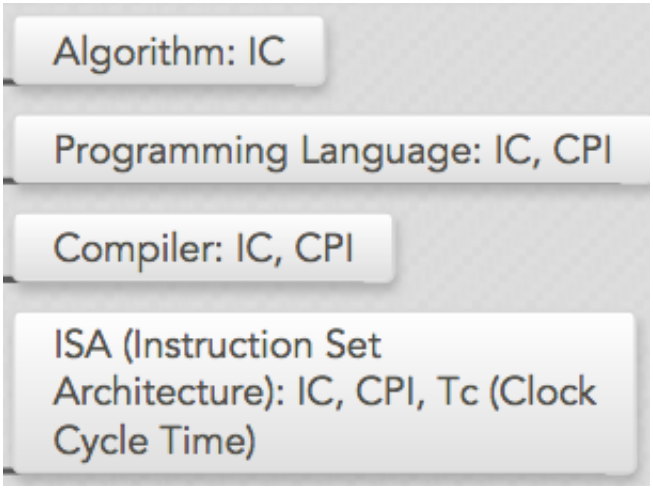
1.2. What are 2 classes of Computer PostPC Era?



1.3. Performance depends on?



1.4. Which software factors that affect the program performance?



1.5. 4 principles of MIPS design

1.6. Range of 2-s Complement Signed Interger

Range: -2^{n-1} to $+2^{n-1} - 1$

- Most Positive: `0111 1111 ... 1111`
- Most Negative: `1000 0000 ... 0000`

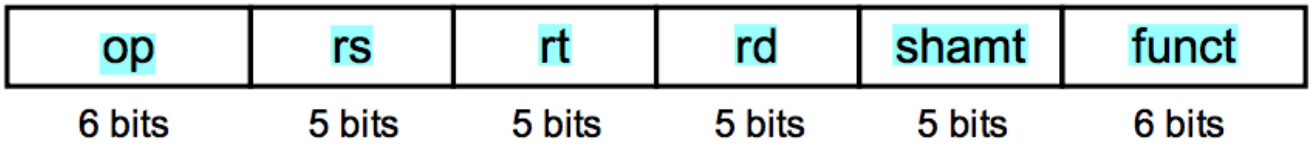
1.7. What are 32 MIPS registers?

Register Name	Common Name	Description
$0	zero	0
$1	at	Assembler Temporary
$2-$3	v0-v1	Result register of functions
$4-$7	a0-a3	Arguments
$8-$15	t0-t7	Temporary Registers
$16-$23	s0-s7	Saved registeres for input/output
$24-$25	t8-t9	Temporary Registers
$26-$27	k0-k1	Reserved for OS kernel
$28	gp	Global Pointer
$29	sp	Stack Pointer
$s30	s8	Saved registers
$31	ra	return address

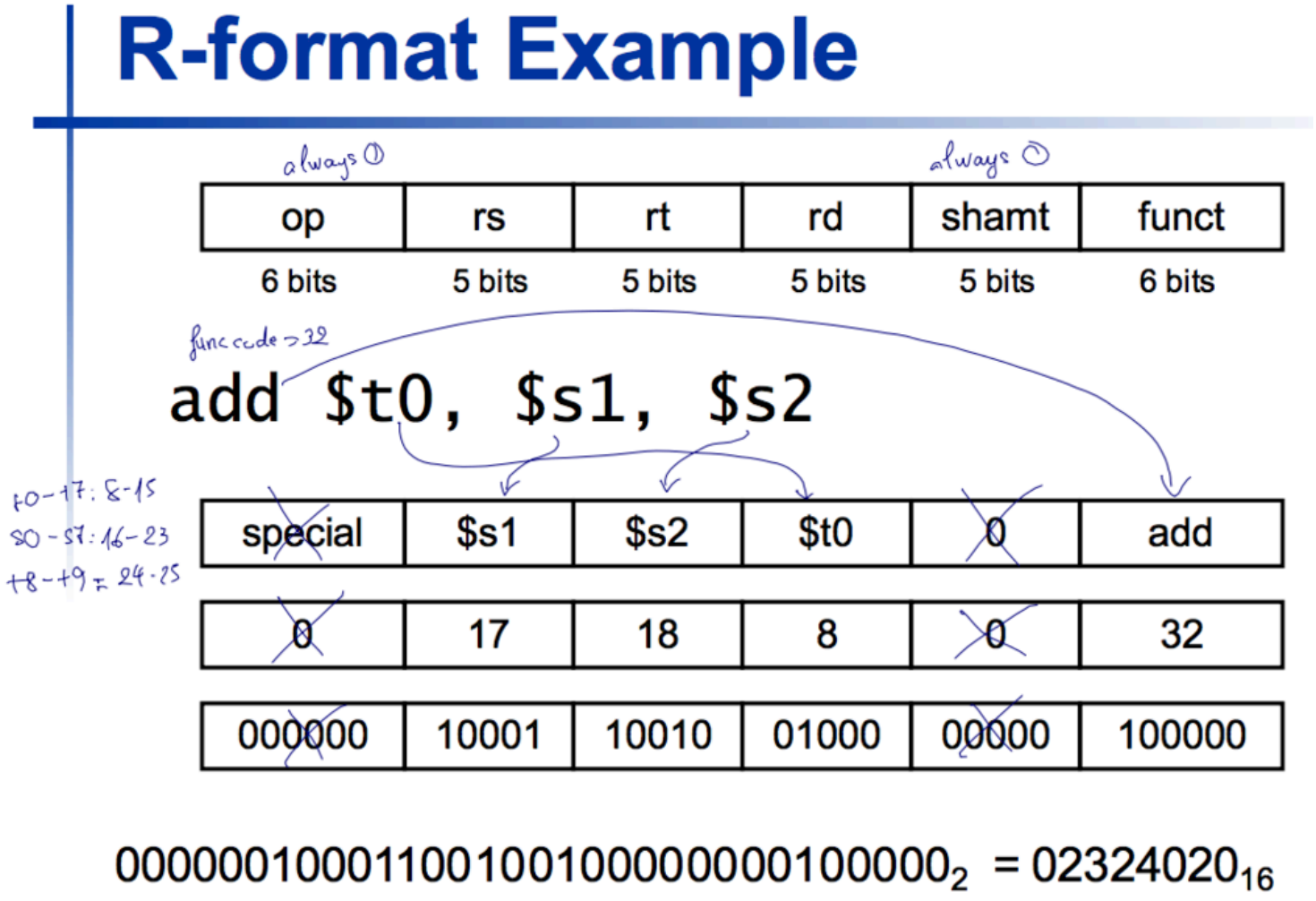
1.8. What is R-type instruction format?

MIPS R-format Instructions

"shift amount"



- Instruction fields
  - op: operation code (opcode, 6 bits) (is 0 for R-type)
  - rs: first source register number (5 bits)
  - rt: second source register number (5 bits)
  - rd: destination register number (5 bits)
  - shamt: shift amount (5 bits; 00000 for now)
  - funct: function code (6 bits; extends opcode)



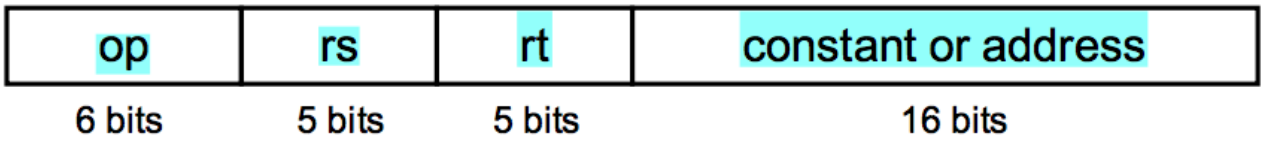
Opcode: 0

Func Code:

add	sub	mult	div	sll	srl	and	or	nor	jr	slt
32	34	24	26	0	2	36	37	39	8	42

1.9. What is I-Type instruction format?

MIPS I-format Instructions



- Immediate arithmetic and load/store instructions
  - rt: destination or source register number
  - Constant: -2^{15} to $+2^{15} - 1$ 16 bit signed number $(-2^{n-1} \to 2^{n-1}-1)$
  - Address: offset added to base address in rs
- `beq rs, rt, L1` ← label on the right (usually dest is leftmost)
  - if (rs == rt) branch to instruction labeled L1;
- `bne rs, rt, L1` ← label on the right (usually dest is leftmost)
  - if (rs != rt) branch to instruction labeled L1;

Func Code: 0

Opcode:

addi	beq	bne	lw	sw
8	4	5	35	43

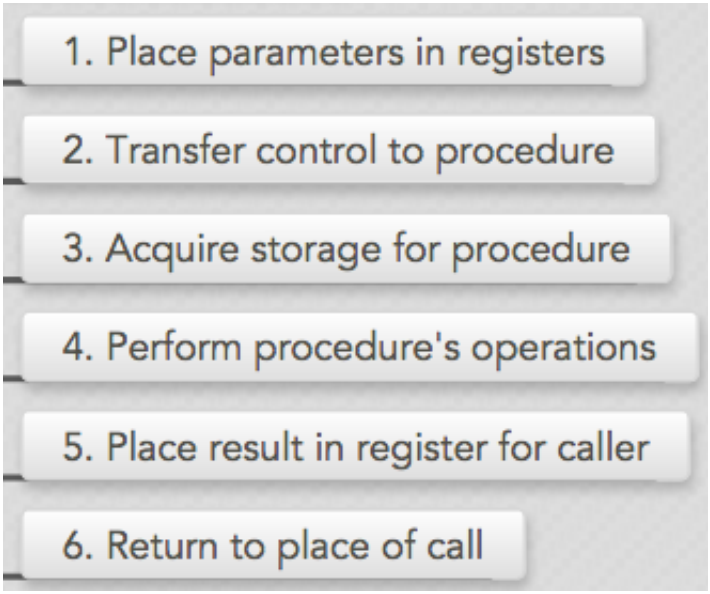
1.10. What is J-Type instruction format?

opcode (6)	target (26)

Opcode:

j	jal
2	3

1.11. What is the steps in procedure calling?



1.12. Size per Character

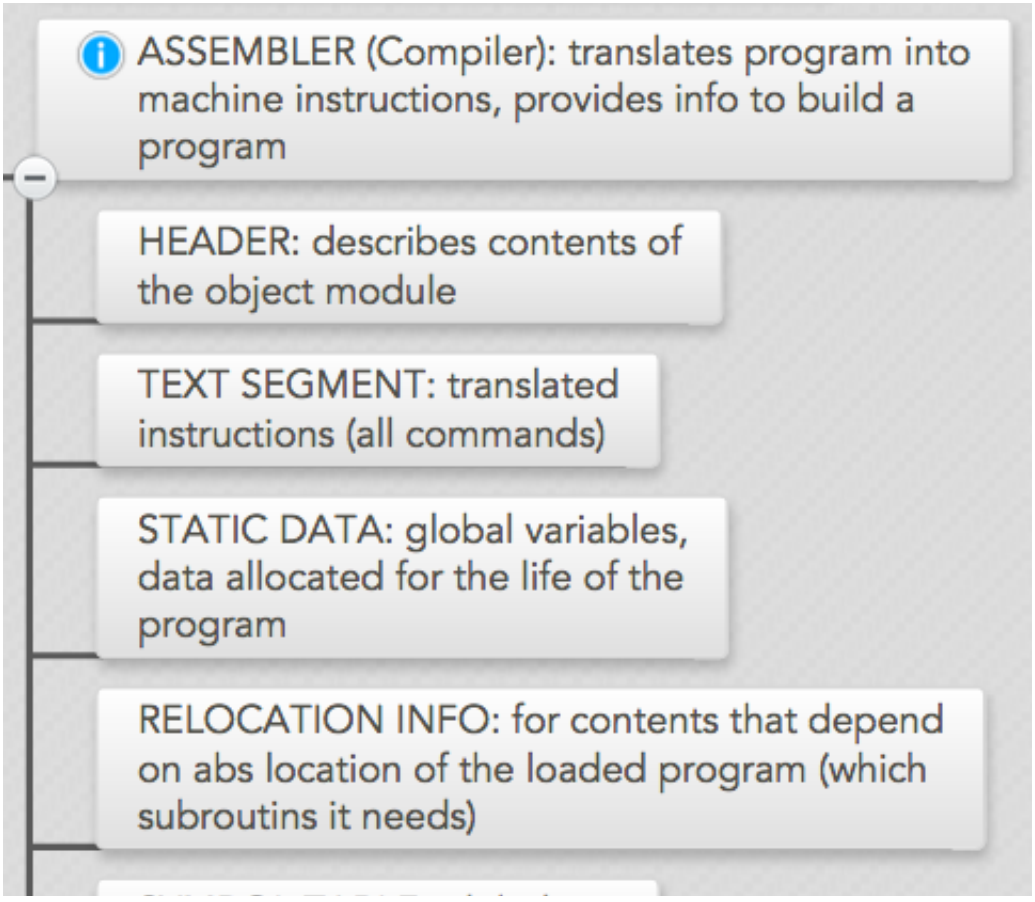
- Byte-Encoded (ASCII, Latin): 1 byte per char
- Java: 2 bytes per char
- Unicode: 4 bytes per char

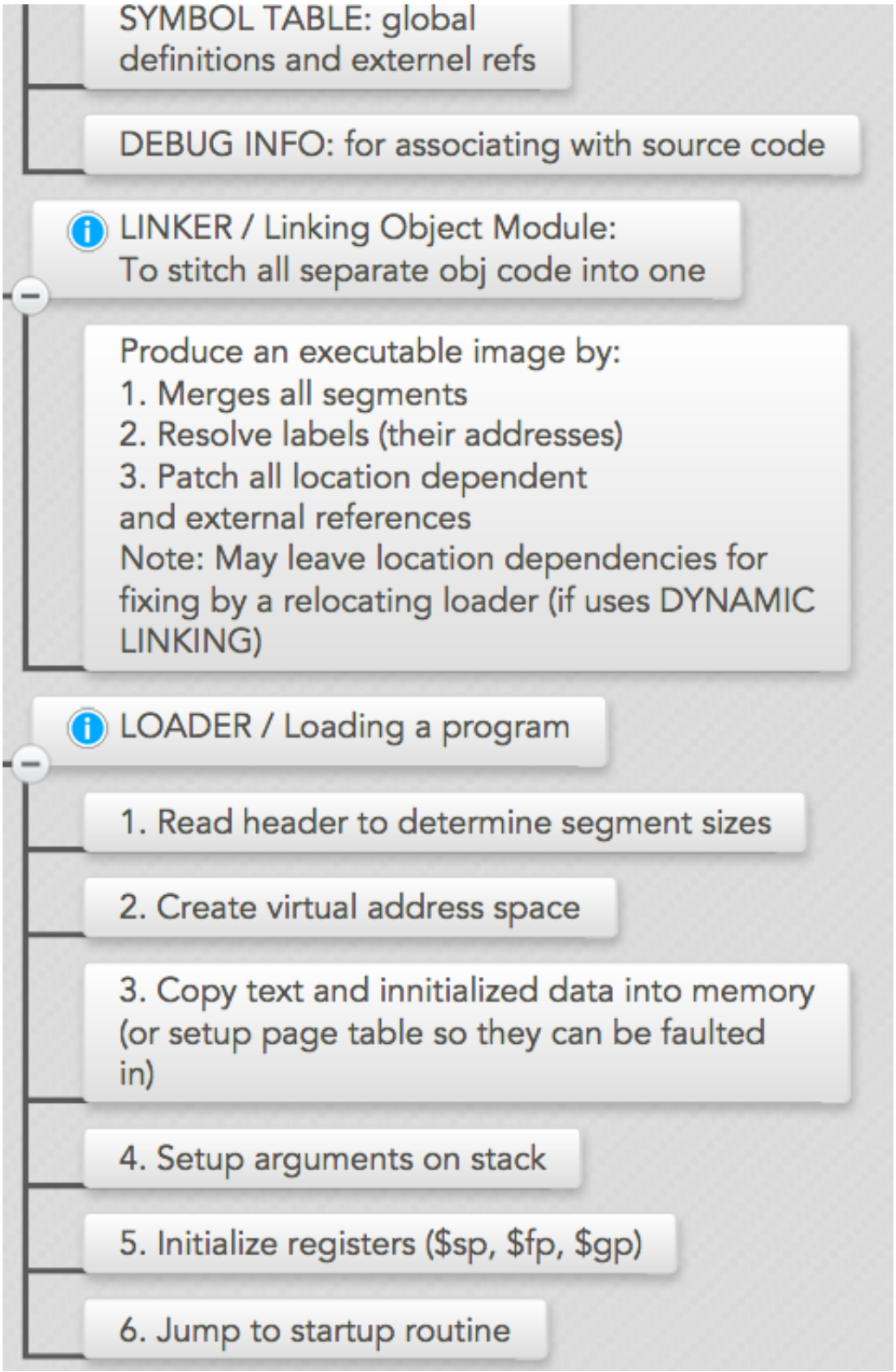
1.13. How to achieve atomic memory update in shared memory using load linked (ll) and store conditional (sc)?

Load linked: ll rt, offset(rs)
Store conditional: sc rt, offset(rs)

Load linked (LL) and store conditional (SC) instructions are a way to achieve atomic memory updates in shared memory multiprocessor systems, without locking memory locations for exclusive access by one processor.
The idea is that you use LL to load the value stored at a memory location into a register, modify it however you like there, and subsequently write it back to the same place using SC. SC will only overwrite the value in memory with your modified one if no other processor has altered it while you were working on the copy in the register. It has the side-effect of setting a status flag to indicate whether or not it was successful.
When the updated value is successfully stored, a thread can trust that its read-modify-write sequence was completed without interference from other threads. On a failure, it is up to the program to decide whether to give up or reload the address and try again, but at least it doesn't produce an undetected Race condition.

1.14. What is the code translation & startup process in MIPS





1.15. What is the differences between Array and Point implementation in MIPS

Array	Pointer
Must have "multiply" inside the loop to recalculate the address	only "add" is required
require base address, and index of current step	require only current location (stored in the pointer)

1.16. MIPS Fallacies

- Powerful instruction leads to high performance is NOT TRUE
- Using assembly code to achieve high performance is NOT TRUE
- Instruction set of MIPS doesn't change (i.e. Backward compatibility) is NOT TRUE

2. Practical Questions

2.1. Performance Formula

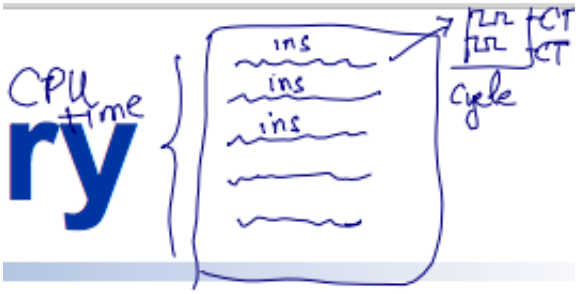
$$Performance = \frac{1}{ExecutionTime}$$

Expand **Clock Cycle Time** => $ClockCycleTime = \frac{1}{ClockRate}$

$$CPUTime = ClockCycles * ClockCycleTime = \frac{ClockCycles}{ClockRate}$$

Expand **Clock Cycle** => $ClockCycles = InstructionCount * CyclesPerInstruction$

$$CPUTime = InstructionCount * CPI * ClockCycleTime = \frac{InstructionCount * CPI}{ClockRate}$$



$$CPUTime = \frac{Instructions}{Program} \frac{ClockCycles}{Instruction} \frac{Seconds}{ClockCycle}$$

2.2. The Power Consumption Forumla

$$Power = Capacity * Voltage^2 * Frequency$$

2.3. The Amdahl's Law Formula

$$T_{improved} = \frac{T_{affected}}{improvementFactor} + T_{unaffected}$$

2.4. How to calculate 2-s complement number?

1. Invert the digits in positive binary form.

   $2_{10} = 0010_2$, inverted: 1101_2

2. Then add 1

   $1101_2 + 1_2 = 1110_2 = -2_{10}$

Repeat the same process to convert 2-s negative number to positive number.