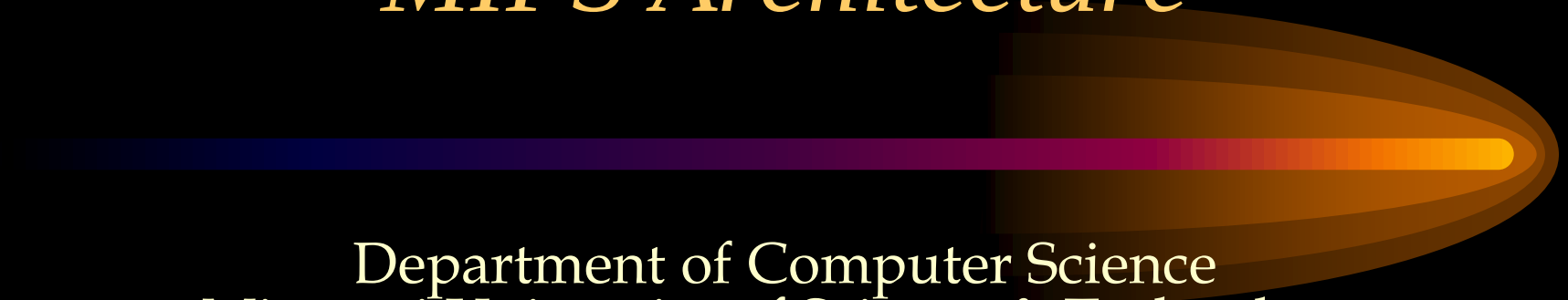
Computer Organization

MIPS Architecture



Department of Computer Science
Missouri University of Science & Technology
hurson@mst.edu

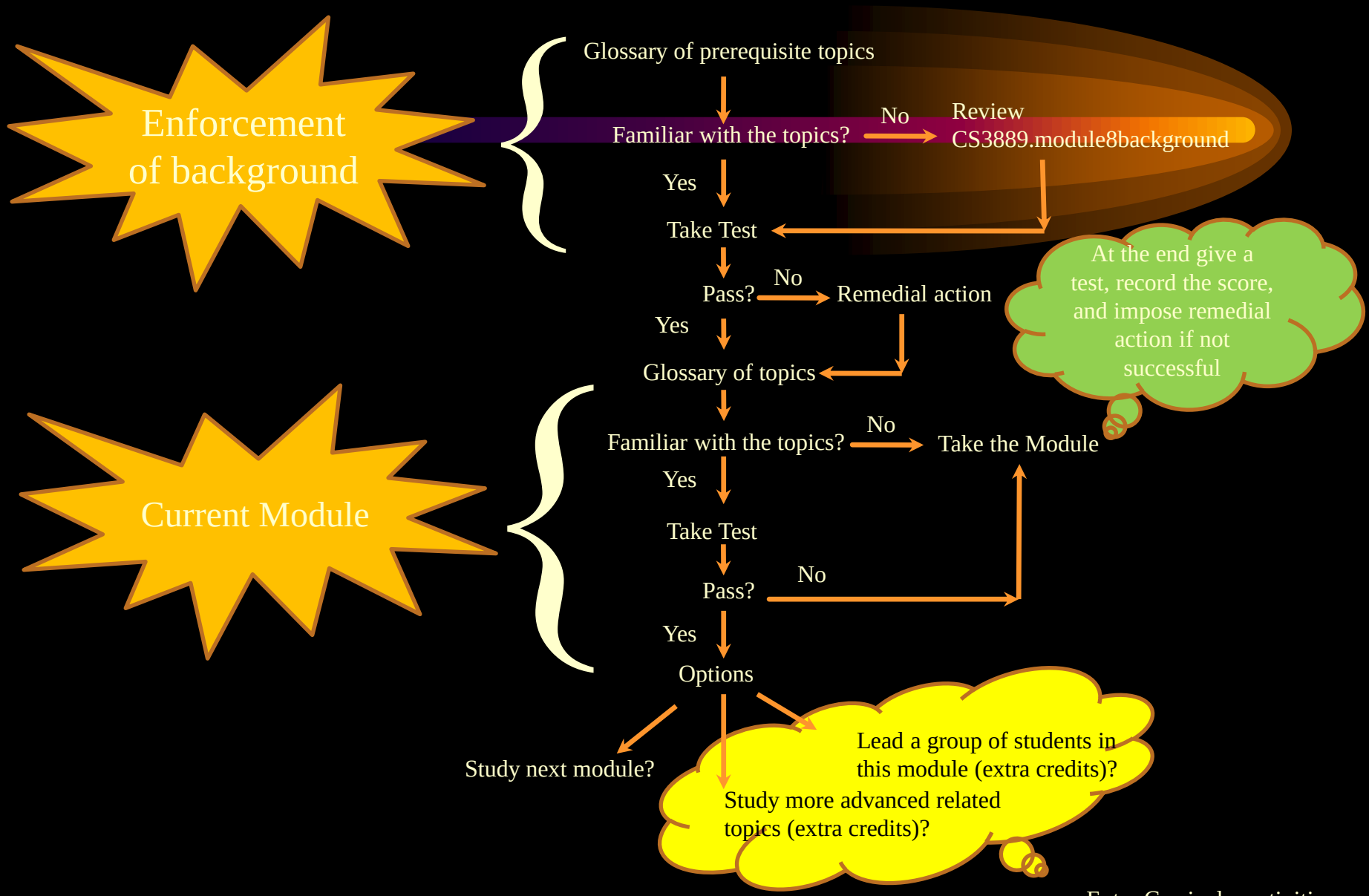
Computer Organization

Note, this unit will be covered in three lectures. In case you finish it earlier, then you have the following options:

1) Take the early test and start CS3889.module9
2) Study the supplement module (supplement CS3889.module88)
3) Act as a helper to help other students in studying CS3889.module8

Note, options 2 and 3 have extra credits as noted in course outline.

Computer Organization



MIPS Organization

- MIPS — Microprocessor without Interlocked Pipeline Stages
  - MIPS processors (R2000 and R3000) have 5-stage pipeline, R4000 and R4400 have 8-stage pipeline, and the number of pipeline stages for R10000 varies, based on the functional units through which the instruction must pass:
    - Integer instructions 5 stages
    - Load/Store 6 stages
    - Floating point 7 stages.

MIPS Organization

- MIPS – Microprocessor without Interlocked Pipeline Stages
  - First MIPS instruction set architecture was MIPS I followed by MIPS II-MIPS V.
  - The current MIPS instruction set architecture is referred to as MIPS32 (for 32-bit architecture) and MIPS64 (for 64-bit architecture).
  - MIPS32 has 168 32-bit instructions.

MIPS Organization

- General Configuration
  - A word addressable, 3-address machine
  - A Load/Store instruction set
  - Register Mode Operations
  - 32 32 bits registers
  - Byte Addressable Main memory
  - Main memory is of size $2^{30} * 32$
  - Fixed instruction length of 32 bits

MIPS Organization

- General Configuration
  - MIPS Supports:
    - Register,
    - Base or displacement (Index)
    - Immediate
    - PC relative, and
    - Pseudo direct addressing modes
  - MIPS Supports 3 different instruction formats:
    - R (Register) Type
    - I (Immediate) Type
    - J (Jump) Type

MIPS Organization

- Register Configuration

Name	Register no.	Usage
$zero	0	Constant zero
$$v_0$-$v_1$	2-3	Return values
$$a_0$-$a_3$	4-7	Input parameters
$$t_0$-$t_7$	8-15	Temporary Values
$$s_0$-$s_7$	16-23	Saved Values
$$t_8$-$t_9$	24-25	Temporary Values
$gp	28	Global pointer
$sp	29	Stack pointer
$fp	30	Frame pointer
$ra	31	Return address

MIPS Organization

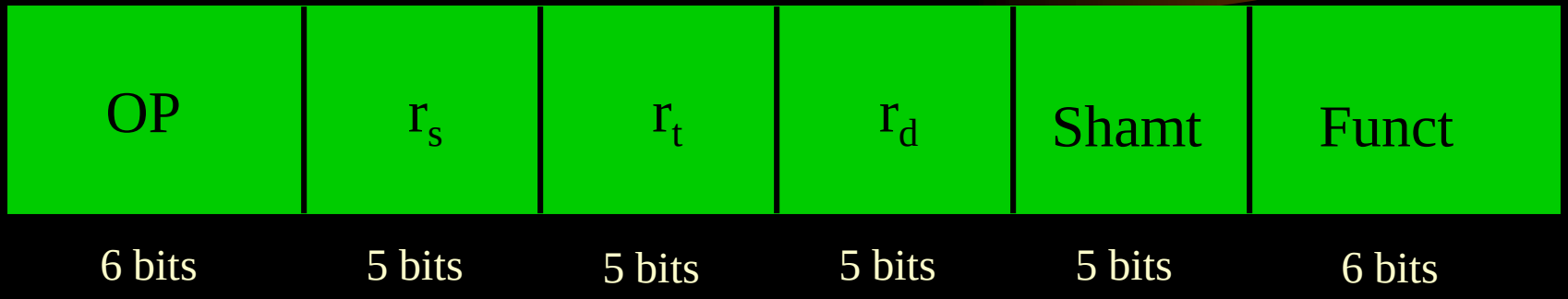
- Register Configuration
  - Register zero (r_0) is hard-wired to a value of zero,
  - r_{31} is the default register for use with certain instructions i.e., it is used as implied mode,
  - r_1 is reserved, r_{26} and r_{27} are used by the operating system.

MIPS Organization

- Addressing Modes
  - Register addressing: operand is a register
  - Base or displacement (Index): operand is at the memory location whose address is the sum of a register and a constant specified in the instruction
  - Immediate: operand is a constant defined in the instruction
  - PC relative: the address is the sum of the PC and a constant defined in the instruction
  - Pseudo direct addressing: the address is the 26 bits of the value defined in the instruction concatenated with the upper 4 bits of PC.

MIPS Organization

Instruction Formats

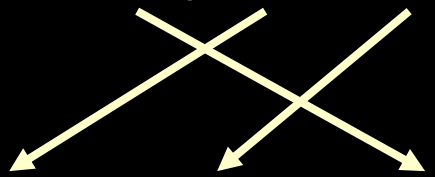


- OP: Basic operation code: op code
- r_s: 1st source register
- r_t: 2nd source register
- r_d: destination register
- Shamt: Shift amount
- Funct: Function code: modifier to op code.

MIPS Organization

- Instruction Formats
  - R-Type Instructions (Example)

add t_0, s_1, s_2

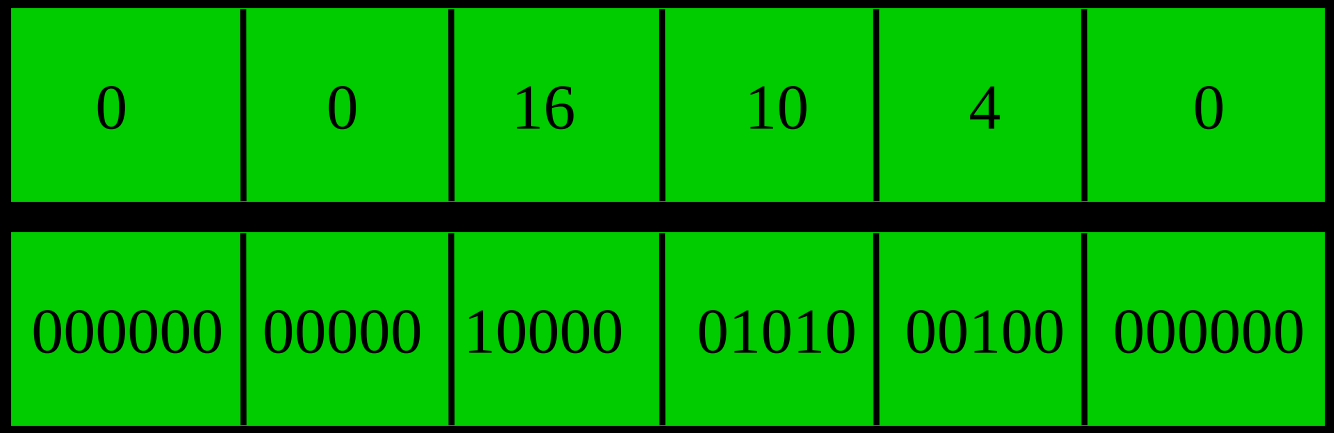


0	17	18	8	0	32
000000	10001	10010	01000	00000	100000

MIPS Organization

- Instruction Formats
  - R-Type Instructions (Example)

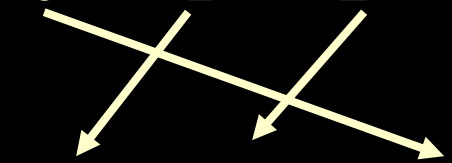
sll t_2, s_0, 4 (Shift s_0 left 4 positions to t_2)



MIPS Organization

- Instruction Formats
  - R-Type Instructions (Example)

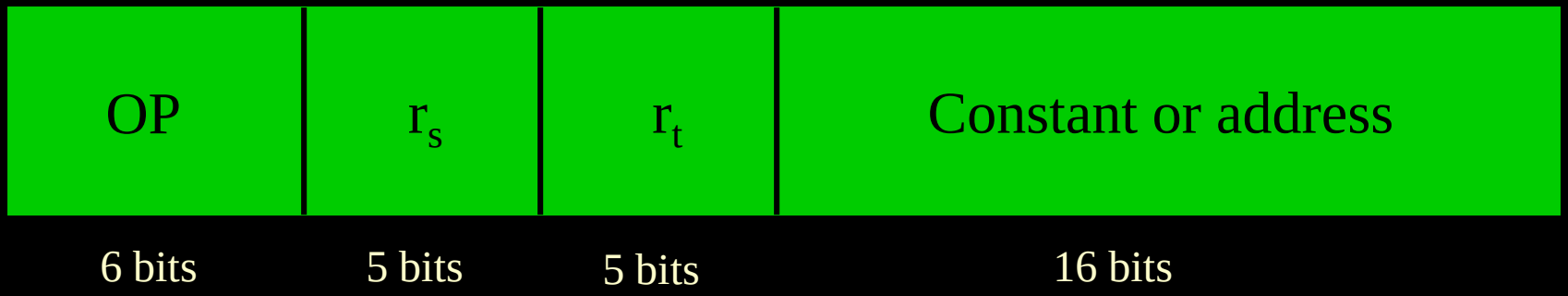
and $\$t_0$, $\$t_1$, $\$t_2$ (and $\$t_1$ with $\$t_2$ into $\$t_0$)



0	9	10	8	0	24
000000	10000	10001	01010	00000	011000

MIPS Organization

Instruction Formats



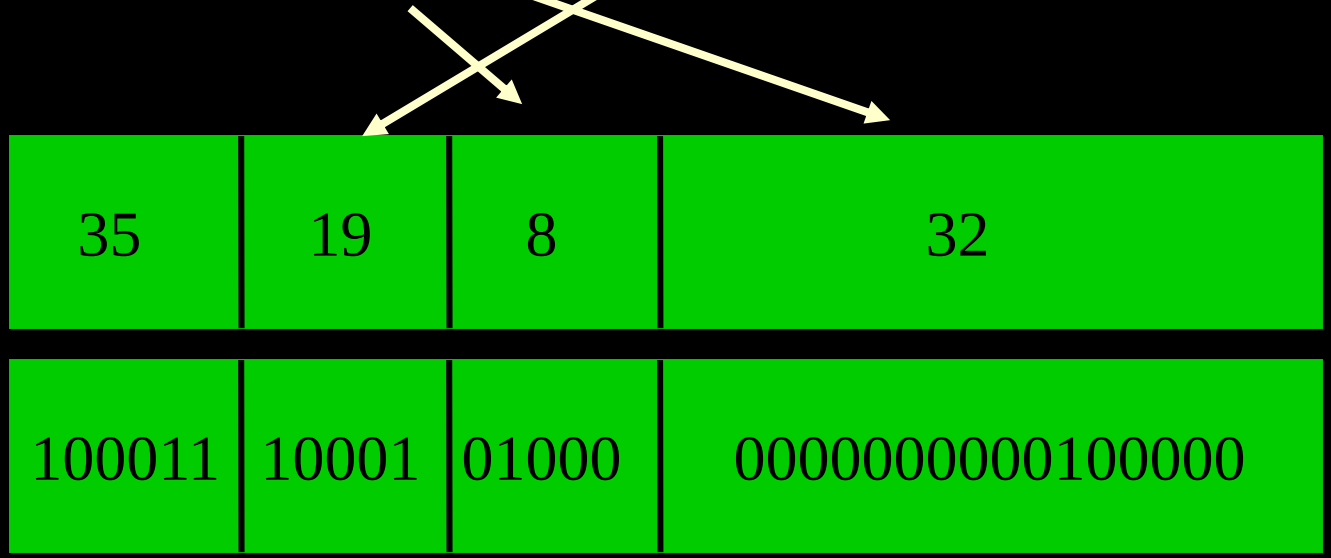
- OP: Basic operation code: op code
- r_s: base (index) register
- r_t: source/destination register

MIPS Organization

- Instruction Formats
  - I-Type Instructions (Example)

Assume the base address of the array A is in $\$s_2$.

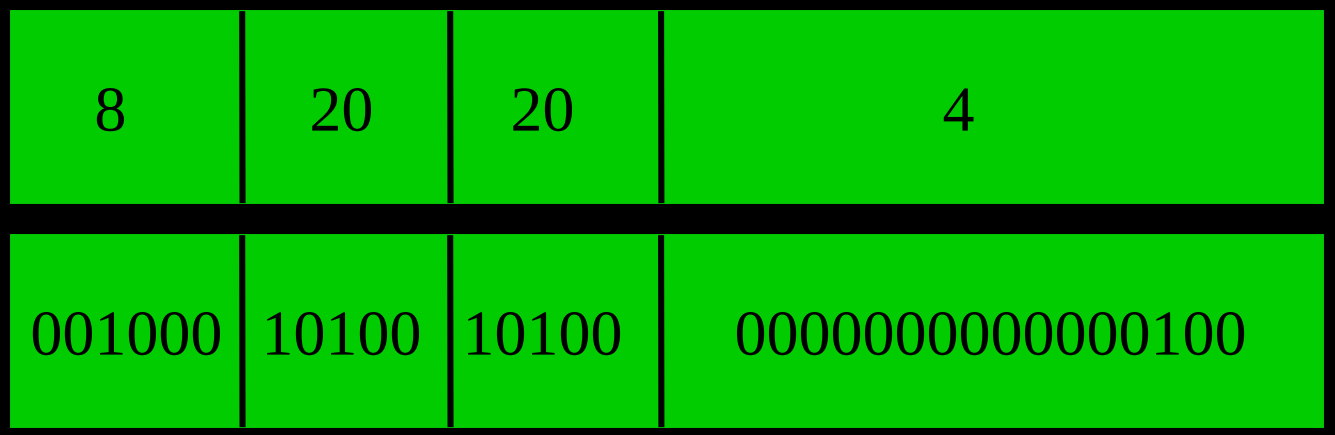
lw $\$t_0$, 32($\$s_2$) (load word A[8] into $\$t_0$)



MIPS Organization

- Instruction Formats
  - I-Type Instructions (Example)

addi s_3$, s_3$, 4 (add constant 4 to s_3)



MIPS Organization

- Instruction Formats – Assembly code
  - If t_1 has the base for the array A and s_2 corresponds to h, then write an assembly code for:

    A[300] = h + A[300]

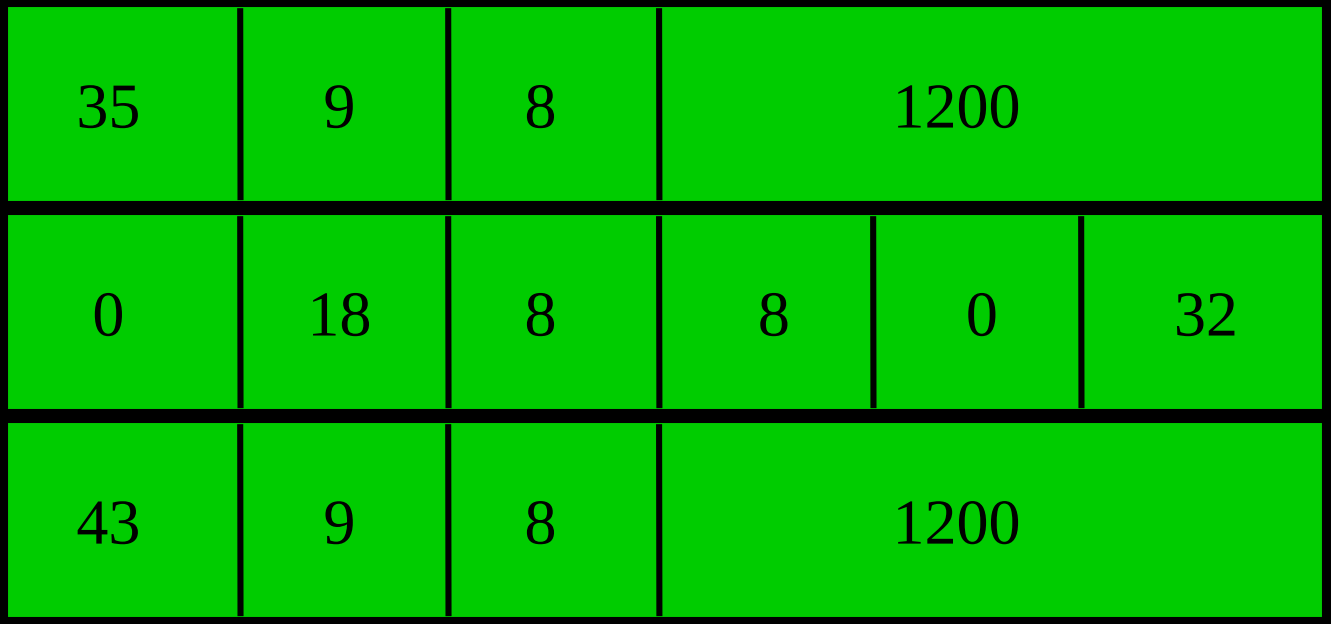
    lw t_0, 1200(t_1)

    add t_0, s_2, t_0

    sw t_0, 1200(t_1)

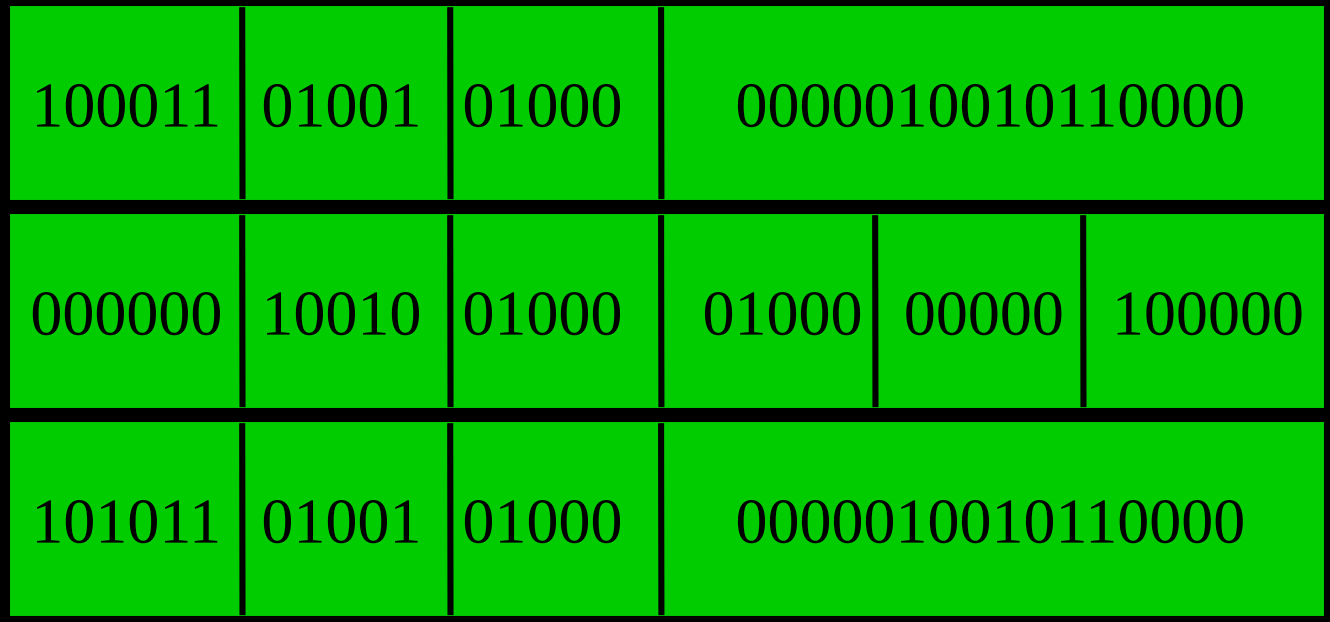
MIPS Organization

- Instruction Formats – Machine code



MIPS Organization

- Instruction Formats – Machine code (Binary)



MIPS Organization

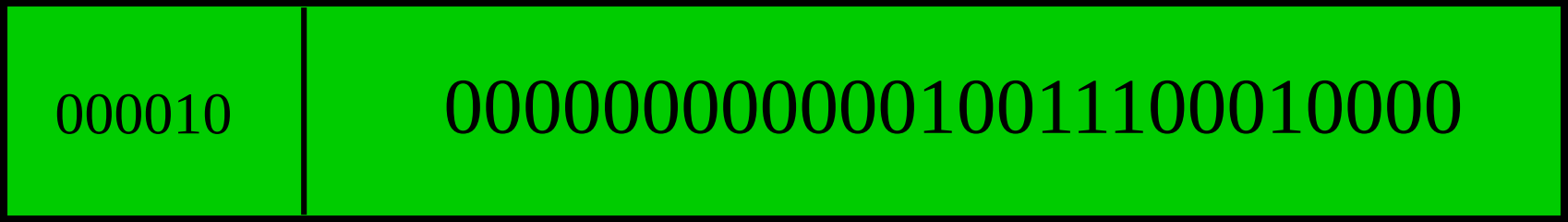
- Instruction Formats
  - J-Type Instructions

- OP: Basic operation code: op code

MIPS Organization

- Instruction Formats
  - J-Type Instructions (Example)

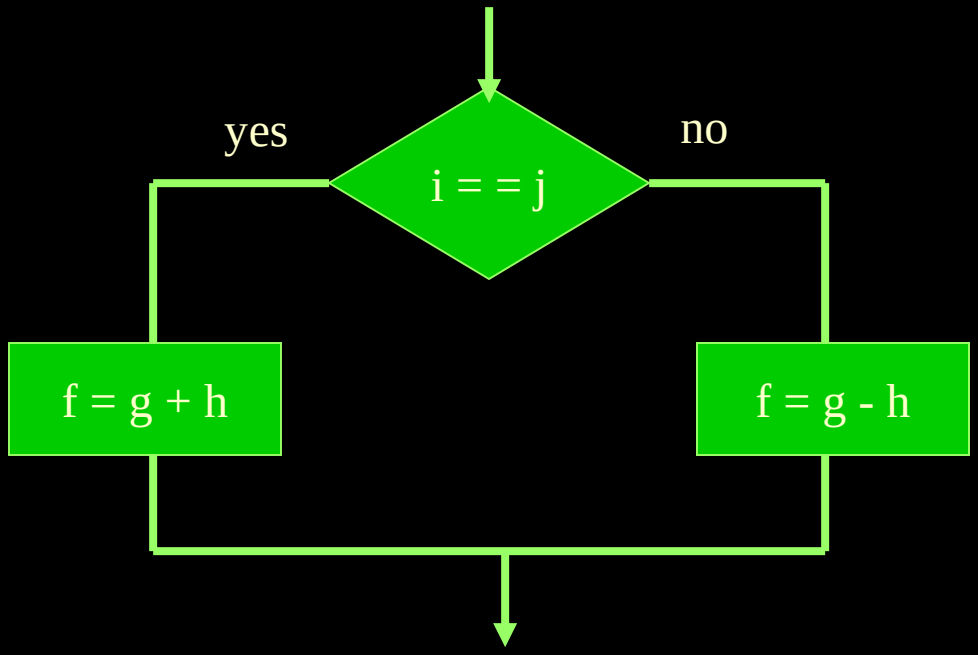
    goto 2500



MIPS Organization

- Instruction Formats
  - Instructions for making decision (If-then-else)

    If (i = = j) f = g + h; else f = g – h;



MIPS Organization

Instruction Formats

- If (i = = j) f = g + h; else f = g – h;
- Assuming *i* is in S_3, *j* is in S_4, *g* is in S_1, and *h* is in S_2.

```
        bne   $s3, $s4, Else     goto Else if i ≠ j
        add   $s0, $s1, $s2      f = g + h
        j     Exit
Else:   Sub   $s0, $s1, $s2      f = g – h
Exit:
```

MIPS Organization

- Instruction Formats
  - Instructions for Loops

    While (Save[i] = = k)

        i += 1

    Assume *Save* is an array whose base is in $\$s_6$. Also assume *i* and *k* correspond to registers $\$s_3$ and $\$s_5$, respectively;

MIPS Organization

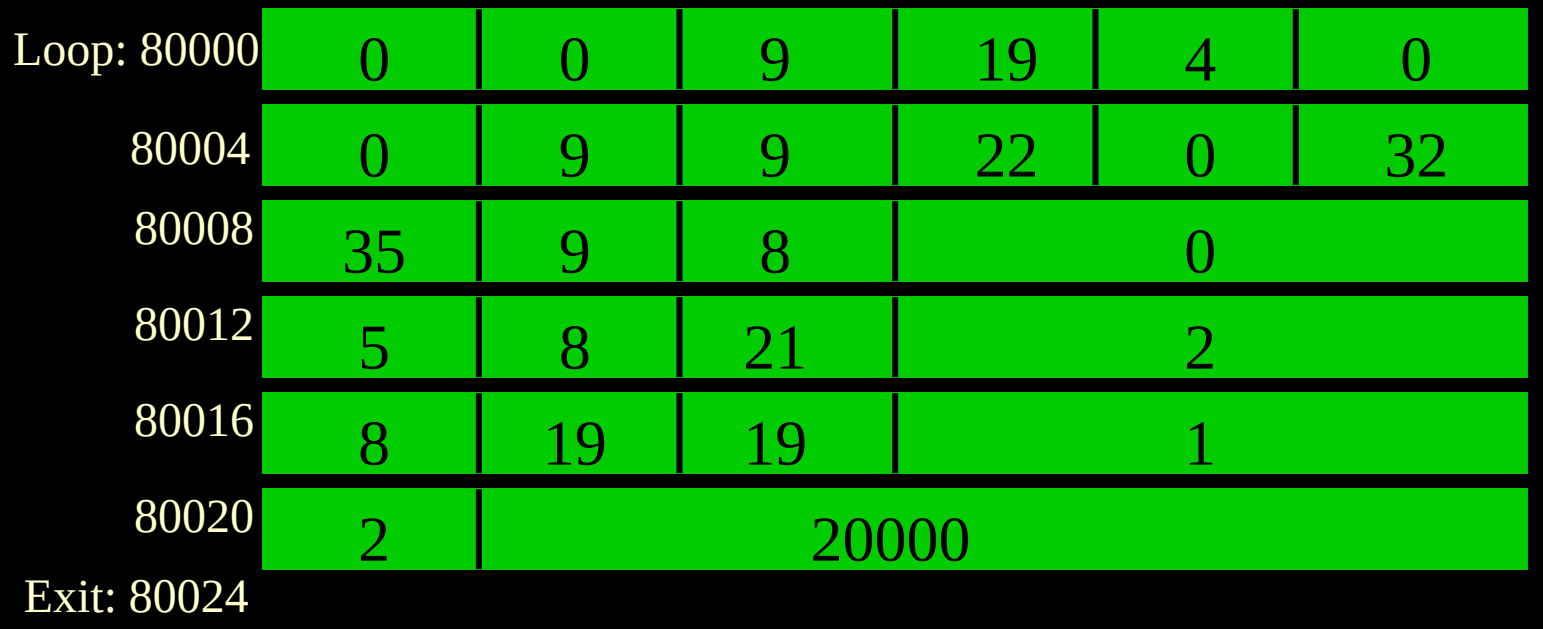
- Instruction Formats

```
Loop:  sll   $t1, $s3, 2      Multiply i by 4,
       add   $t1, $t1, $s6    Calculate the effective address
       lw    $t0, 0($t1)      $t0 = Save[i]
       bne   $t0, $s5, Exit   If Save[i] ≠ k goto Exit
       addi  $s3, $s3, 1      i = i + 1
       j     Loop             Repeat the loop
Exit:
```

MIPS Organization

Instruction Formats

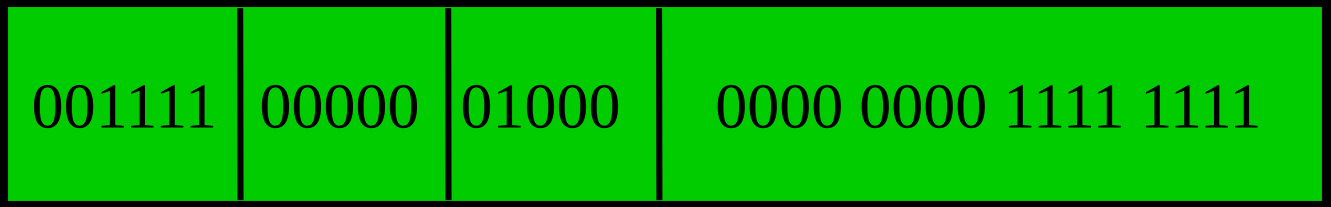
- In the previous example, if the loop is starting at location 80,000, then what is the MIPS machine code:



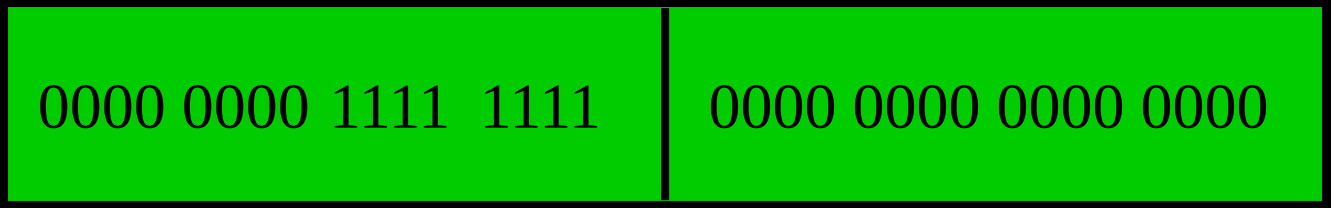
MIPS Organization

- Example
  - Load Upper Immediate (lui) sets the upper 16 bits of a constant in a register

    lui $\$t_0$, 255



  - After execution $\$t_0$ is:



MIPS Organization

- Example
  - Set register $\$s_0$ to

0000 0000 0011 1101 0000 1001 0000 0000

lui $\$s_0$, 61

ori $\$s_0$, $\$s_0$, 2304

MIPS Organization

- Procedure
  - A procedure is a sequence of instructions that performs a specific task based on its input parameters.
  - In the execution of a procedure, the program performs the following steps:
    - Place input parameters in a place where it is accessible by the procedure,
    - Transfer control to the procedure,
    - Acquire the storage resources for the procedure,
    - Execute the procedure,
    - Place the result in a place accessible to the calling program,
    - Return control to the point of origin.

MIPS Organization

- Procedure
  - In MIPS:
    - a0-a3 are used to pass parameters,
    - v0-v1 are used to return values,
    - ra is used to return to the point of origin.
    - The Jump-and-link instruction (jal) saves the return address and jump to the define address:

      jal Procedureaddress
    - The Jump-register (jr) instruction returns the control to the point of origin:

      jr ra

MIPS Organization

- Procedure
  - Assume the following subroutine was called:

```
Int leaf_example (int g, int h, int i, int j)
    Int f;
    F = (g + h) – (I + j)
    Return f;
```

  - The parameters g, h, i, and j are input parameters and f is the return value.
  - If this program is compiled for MIPS then we have:

MIPS Organization

- Procedure
  - The parameters g, h, i, and j correspond to $\$a_0$-$\$a_3$, and f corresponds to $\$s_0$.
  - The compiled program starts with the label "leaf_example".
  - Internally the procedure will use some of the registers (working registers) during the execution time, as a result, the contents of these registers must be saved and restore after the execution of the subroutine. This will be done in the system stack.

MIPS Organization

- Procedure
  - addi $sp, $sp, -12 # The first four instructions allow us to save
  - sw $$t_1$$, 8($sp) the contents of the working registers in the
  - sw $$t_0$$, 4($sp) stack#
  - sw $$s_0$$, 0($sp)
  - add $$t_0$$, $$a_0$$, $$a_1$$
  - add $$t_1$$, $$a_2$$, $$a_3$$
  - sub $$s_0$$, $$t_0$$, $$t_1$$
  - add $$v_0$$, $$s_0$$, $zero
  - lw $$s_0$$, 0($sp) #This instruction and the next three instructions
  - lw $$t_0$$, 4($sp) are intended to restore the original contents of
  - lw $$t_1$$, 8($sp) the working registers before control transfers
  - addi $sp, $sp, 12 to the callee procedure#
  - jr $ra #This instruction transfers the control back to callee#

Instruction Cycle

- An instruction cycle is implemented in five basic steps:

Instruction Fetch — IF

$IR \leftarrow Mem[PC]$

$NPC \leftarrow PC + 4$

Instruction decode and register fetch — ID

$A \leftarrow Regs[IR_{6,..,10}]$

$B \leftarrow Regs[IR_{11,..,15}]$

$Imm \leftarrow ((IR_{16})^{16} \,\#\#\, IR_{16,..,31})$

Instruction Cycle

- Execution and effective address calculation — EX

Memory reference

ALUoutput ← A + Imm

Reg.-Reg. ALU instruction

ALUoutput ← A op-code B

Reg.-Immediate ALU instruction

ALUoutput ← A op-code Imm

Branch

ALUoutput ← NPC + Imm,
Cond ← (A op-code 0)

Computer Organization

Instruction Cycle

- Memory access/branch completion — MEM

Memory reference

LMD ← Mem [ALUoutput] or

Mem [ALUoutput] ← B

Branch

If (cond.) PC ← ALUoutput else PC ← NPC

Computer Organization

- Instruction Cycle
  - Write Back — WB

Reg.-Reg. ALU instruction

$$\text{Regs}[IR_{16,..,20}] \leftarrow \text{ALUoutput}$$

Reg.-Immediate ALU instruction

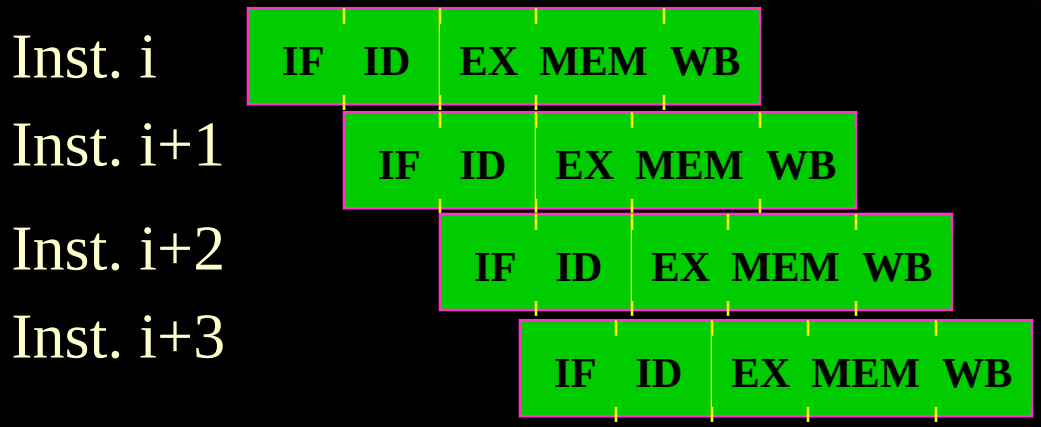
$$\text{Regs}[IR_{11,..,15}] \leftarrow \text{ALUoutput}$$

Load instruction

$$\text{Regs}[IR_{11,..,15}] \leftarrow \text{LMD}$$

Computer Organization

- Pipelined instruction cycle



A pipelined instruction cycle gives a peak performance of one instruction every step.