EE 3755

Handout entitled

Third example computer: The MIPS computer.

The MIPS computer is pretty similar to the second example computer except:

(1) Register R0 always contains zero; (you can only read it but not write it).

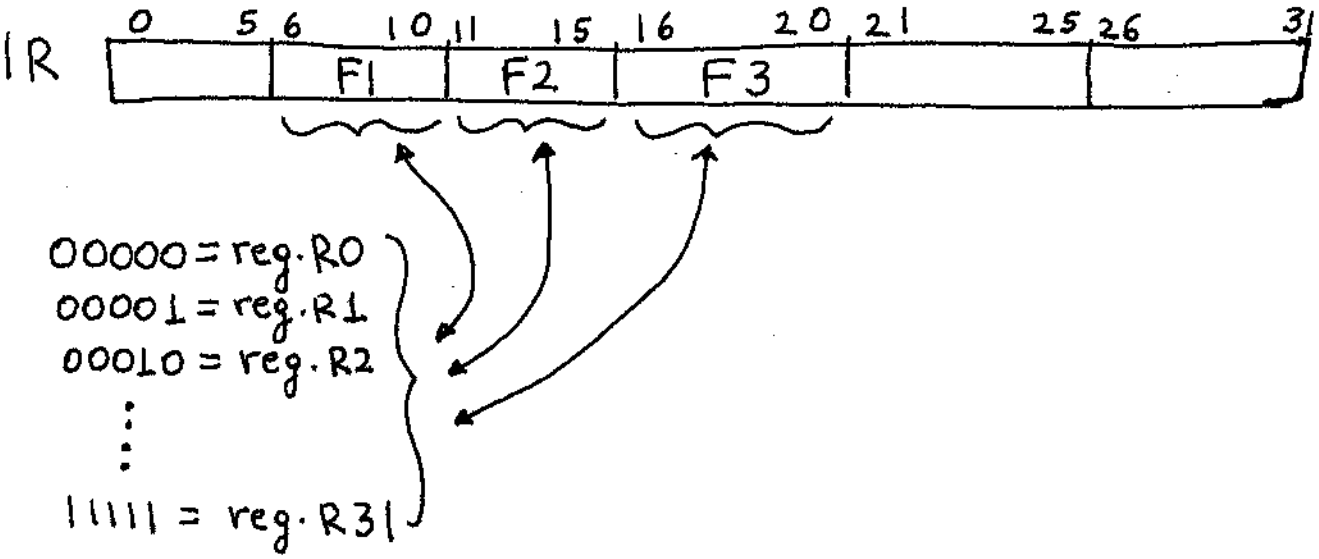
(2) Its instruction set architecture contains an instruction "slt" (set on less than).

In assembly language the "slt" instruction looks like

```
slt $t0, $s3, $s4
```

→ Meaning: Register $t0 is set to 1 if the value in register $s3 is less than the value in register $s4. Otherwise, register $t0 is set to 0.

In machine language the "slt" instruction looks like



Meaning: Register specified by field F3 is set to 1 if the value in the register specified by field F1 is less than the value in the register specified by the field F2. Otherwise, register specified by field F3 is set to 0.

Question:

Write in AHPL the control steps for the "slt" instruction of the MIPS computer.

600 BBUS=(R0!R1!....!R31)*DCDA(IR[11:15]);
OBUS=BBUS; MD←OBUS; "Transfer register specified by field F2 in MD"

601 ABUS=$\overline{MD}$; BBUS=(R0!R1!...!R31)*DCDA(IR[6:0]);
cin=1; cff←ADD[0](ABUS; BBUS; cin).

602 (R0!R1!....!R31)*DCDA(IR[16:20])←($\overline{32T0}$! 32T0)
*($\overline{cff}$, cff); →(1).

Note: The above solution assumed that the registers contain unsigned numbers. What about if the numbers were signed?

(3) Its instruction set architecture does not contain the branch instruction of the "first example computer" which is the same as the branch instruction of the "second example computer".

Instead, the MIPS instruction set architecture contains two simpler branch instructions: the "beq" and the "bne".

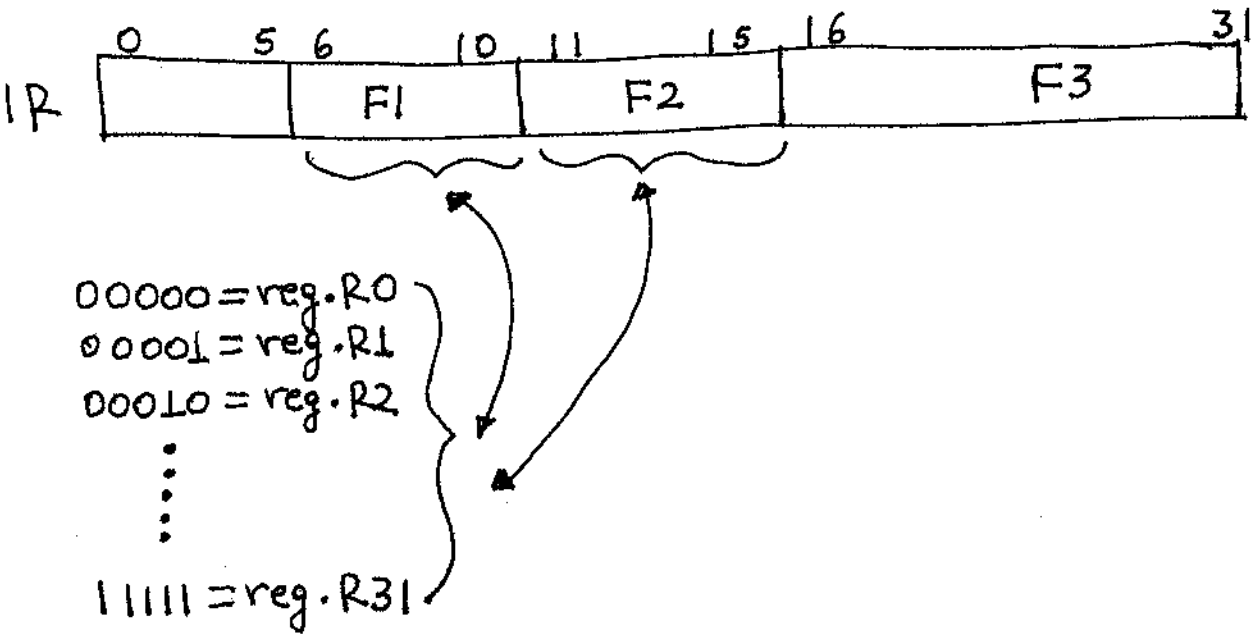
(3.1) The MIPS instruction set architecture contains an instruction "beq" (branch if equal).

In assembly language the "beq" instruction looks like

```
beq register1, register2, L1
```

→ Meaning: Go to statement labeled L1 if the value in register 1 equals value in register 2.

In machine language the "beq" instruction looks like



Meaning: Branch to instruction located at memory address = PC + F3 if (register specified by field F1) = (register specified by field F2). Else, the next instruction is fetched. Here, the 16-bit field F3 = IR[16:31] is a displacement (or offset) and is a signed number. Thus, it must be sign extended.

Question:

Write in AHPL the control steps for the "beq" instruction of the MIPS computer.

Answer:

```
700 BBUS=(R0!R1!....!R31)*DCDA(IR[11:15]);
    OBUS=BBUS; MD ← OBUS. " transfer register specified
                           by field F2 in MD register"
701 ABUS=MD; BBUS=(R0!R1!....!R31)*DCDA(IR[6:10]);
    cin=1; OBUS=ADD[1:32](ABUS; BBUS; cin);
    zff ← V/OBUS.
702 ABUS=(16TO,IR[16:31]!16TO,IR[16:31])*(IR[16], IR[16]);
    BBUS=PC; cin=0;
    OBUS=(ADD[1:32](ABUS;BBUS;cin)!BBUS)*(zff, zff);
    PC ← OBUS; → (1).
```

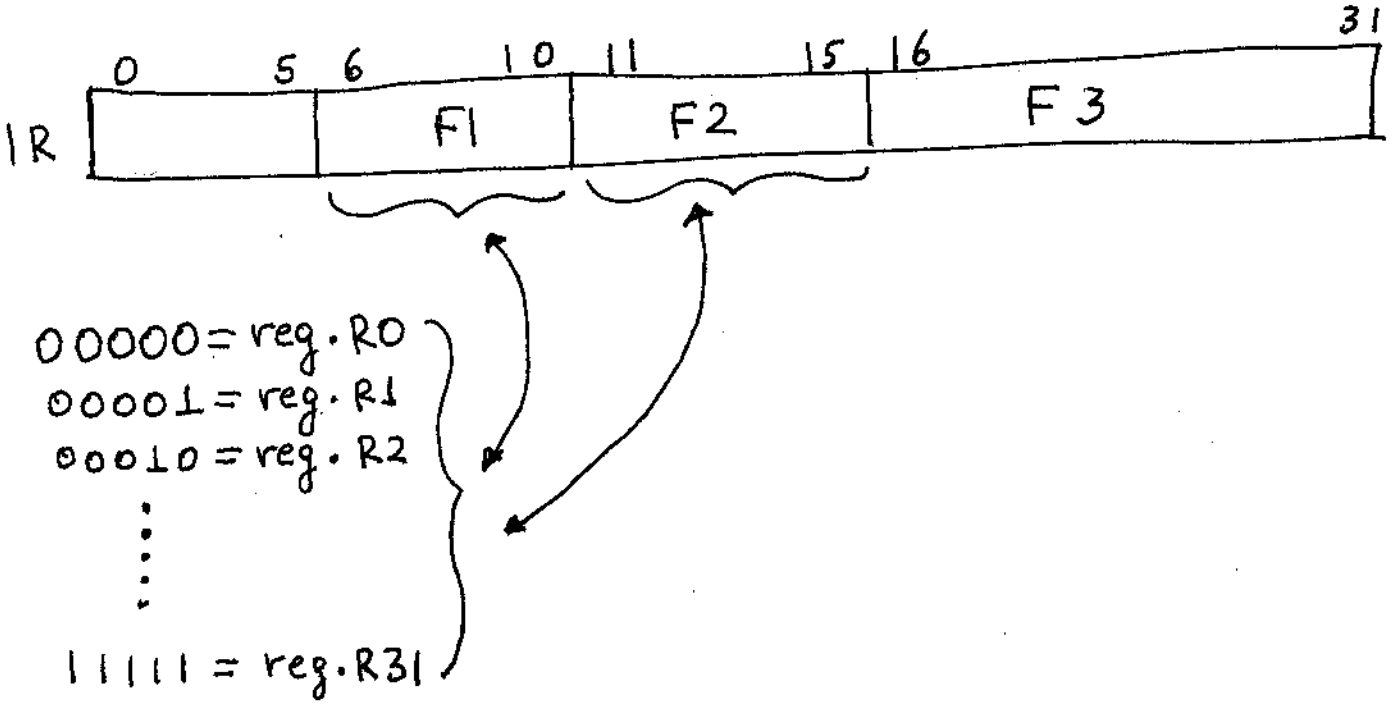
(3.2) The MIPS instruction set architecture contains an instruction "bne" (branch if not equal).

In assembly language the "bne" instruction looks like

```
bne register1, register2, L1
```

→ Meaning: Go to statement labeled L1 if the value in register 1 does not equal the value in register 2.

In machine language the "bne" instruction looks like



Meaning: Branch to instruction located at memory address = PC+F3 if (register specified by field F1) ≠ (register specified by field F2). Else, the next instruction is fetched. Here, the 16-bit field F3=IR[16:31] is a displacement (or offset) and is a signed number. Thus it must be sign extended.

Question:

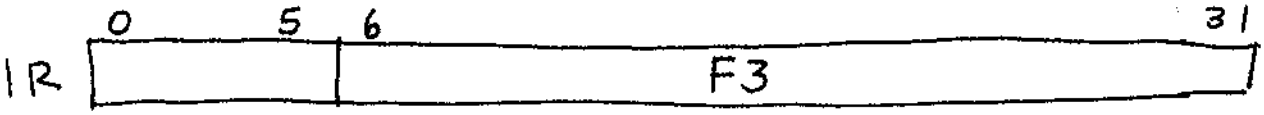
Write in AHPL the control steps for the "bne" instruction of the MIPS computer.

Answer

800 same as step 700 of "beq"
801 " " " 701 " "beq"
802 " " " 702 " "beq" except that

OBUS=(ADD[1:32](ABUS; BBUS; cin)!BBUS)*($\overline{zff}$, zff). Rest is the same.

(4) The MIPS instruction set architecture contains an instruction "jal" (jump and link) shown below:



Meaning: Unconditional jump to subroutine at address = field F3. Save return address in register R31.

Question:

Write in AHPL the control steps for the "jal" instruction of the MIPS computer".

Answer:

900 R31 ← PC

901 PC ← 6TO, IR[6:31]; → (1).

(5) The MIPS computer does <u>not</u> have PUSH and POP instructions.

<u>Question:</u>

Write a MIPS program to PUSH reg. R10 in stack

<u>Answer:</u> In assembly language the program will be

<u>Instruction i:</u> ADDi R29; #-1; R29. " Add to register R29 the #-1 and put result in R29; In other words decrement SP by 1"

<u>Instruction i+1:</u> Store R10; (R29; #0). " Store register R10 at memory address (R29+0) or at memory address = (SP).

In other words for PUSH

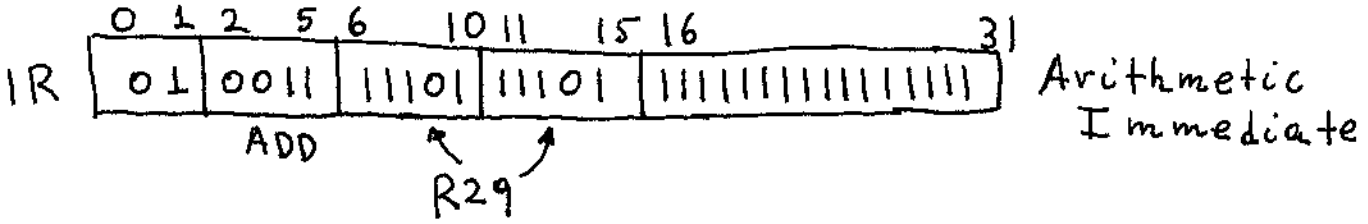
<u>Instruction i:</u> R29 ← R29-1. " Decrement R29 by 1 or decrement SP by 1"

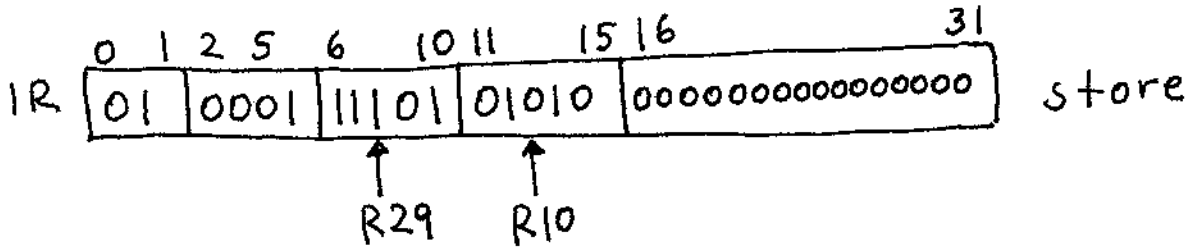
<u>Instruction i+1:</u> Store reg. R10 in memory at address (R29)

↑

contents of R29.

In machine language the program will be





Question:

Write a MIPS program to POP R10 from stack.

Answer:

Instruction i: ADDi R29; #+1; R29. "Add to reg. R29 the #+1 and put result in R29. In other words increment SP by 1"!

Instruction i+1: Load R10; (R29; #-1). " Load reg. R10 with memory data from address (R29-1) or from memory address SP-1.

In other words for POP

Instruction i: R29 ← R29+1. " Increment R29 by 1 or increment SP by 1"

Instruction i+1: Load R10 with memory data from memory address = =((R29)−1)

(6) The MIPS computer does not have register transfer instructions (like the ones on page 11 of the "second example computer"). The register transfer instructions can be accomplished with logic OR (V) and the fact that register R0 always contains zeros.

For example, if we want to do

R1 ← R2

we can do it as

R1 ← R2 V R0

(7) The MIPS does <u>not</u> have load immediate instruction to load 32-bit immediate data to register. It can accomplish this by the load upper immediate instruction and right shift instruction.

(8) The MIPS computer does <u>not</u> have rotate right or left instructions. These can be accomplished by the shift left or right instructions.

<u>For example</u>

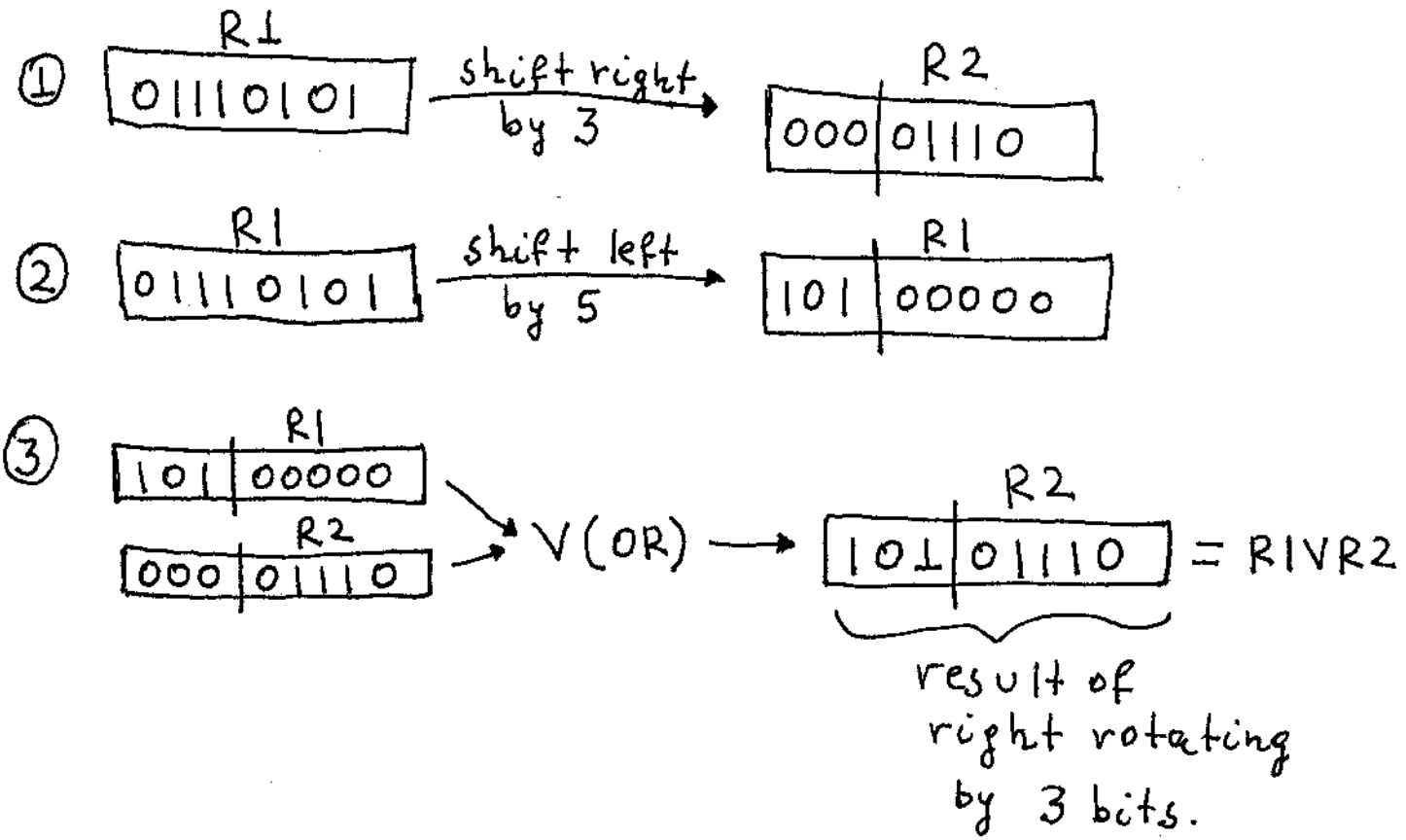
rotate R1 right by k bits and store it in R2 where $k \leq 32 = n$

<u>Answer:</u> $R1 \xrightarrow[\text{by } k]{\text{shift right}} R2$

$$R1 \xrightarrow[\text{by } 32-k \;=\; n]{\text{shift left}} R1$$

$$R1 \vee R2 \longrightarrow R2$$

Example: $n=8$; $k=3$; right direction



⑨ The MIPS computer uses instructions slt, beq, bne and fact that register R0 always contains zeros to create all relative conditions "equal", "not equal", "less than", "less than or equal", "greater than", "greater than or equal" (or $=$; $\neq$; $<$; $\leq$; $>$; $\geq$).

Question:

How do we branch to address if ($S0)<($S1) or in other words how do we accomplish "blt" (branch less than)? Here $S0 and $S1 are two registers.

Answer:

Instruction i: slt $to, $S0, $S1. "Set $to to 1 if ($S0)<($S1). Else set $to to 0.

Instruction i+1: bne $to, R0, address. " Branch to address if ($to)≠0 or if ($S0)<($S1)".

For bgt (branch greater than) $S0, $S1 we do blt $S1, $S0

About ≤ and ≥

Say we want begt (branch on equal or greater than).

- beq address. " If equal go to address; if not
- bgt address. " If greater go to address; if not
- next instruction

etc ----