

SBI Sp 10 2010-01-19

Note Title

1/19/2010

(\* Pre:  $a \leq b+1$  \*)

Pre'  $\rightarrow$

```

i := a; sum := 0;
while i ≠ b+1 do
  sum := sum + A[i];
  i := i + 1;
end;

```

(\* Post:  $sum = \sum_{j=a}^b A[j]$  \*)

$\{a \leq b+1 \wedge i = a \wedge sum = 0\} \Rightarrow sum = \sum_{j=a}^{i-1} A[j]$

Claim:  $sum = \sum_{j=a}^{i-1} A[j]$  is a loop invariant.

~~Check:  $\{sum = \sum_{j=a}^i A[j] \wedge i = b+1\} \Rightarrow sum = \sum_{j=a}^b A[j]?$~~

~~No!~~

Check:  $I \wedge \neg B \Rightarrow Post \vee$

Pre'  $\Rightarrow I$

$0 = \sum_{j=a}^{a-1} A[j]$  ✓

Claim:  $sum = \sum_{j=a}^{i-1} A[j]$  is invariant

(\* Pre:  $a \leq b+1$  \*)

Proof of Claim

By induction on the number of iterations,  $k$

Pre'  $\rightarrow$   $i := a$ ;  $sum := \phi$ ;

while  $i \neq b+1$  do

$sum := sum + A[i]$ ;

$i := i + 1$ ;

end;

Basis:  $k=1$ , i.e.  $I$  holds at the beginning of the first iteration.

We already checked this

(\* Post:  $sum = \sum_{j=a}^b A[j]$  \*)

Ind. step Assume  $I$  holds at the beginning of the  $k$ th iteration.

If the  $k$ -th iteration is the last one, then done. Else,

$i \neq b+1$ , so we need to show that  $\{I \wedge i \neq b+1\} \Rightarrow I$ .

Let  $i$  and  $sum$  be the values of the variables  $i$  and

sum before the  $k$ -th execution of the loop, and let  $i'$  and  $sum'$  be the values of the variables  $i$  and  $sum$  after the  $k$ -th execution of the loop. Then,

$i' = i + 1$  and  $sum' = sum + A[i]$ . So, we need to

Show:  $\left\{ sum = \sum_{j=a}^{i-1} A[j] \wedge i \neq b+1 \right\} \Rightarrow sum' = \sum_{j=a}^{i'-1} A[j]$

$$sum + A[i] = \sum_{j=a}^{i+1-1} A[j]$$

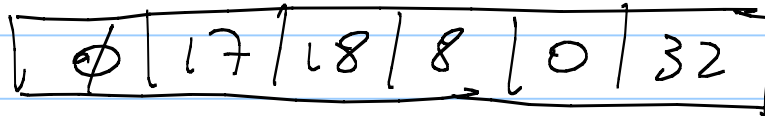
$$= \sum_{j=a}^{i-1} A[j] + A[i]$$



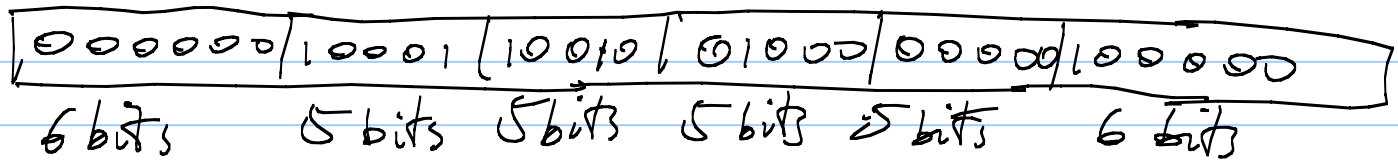
Example (MIPS according to Patterson & Hennessy)

symbolic representation      add \$1, \$2, \$t0

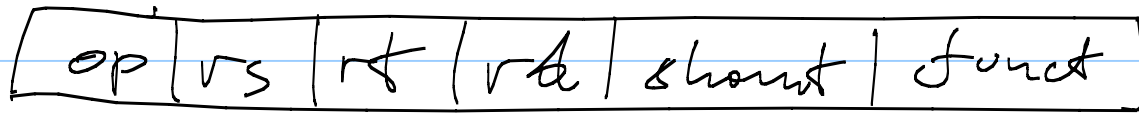
decimal  
representation  
with fields



binary  
representation



instruction  
format



op: opcode (basic operation)

rs: first register source operand

rt: second ~ ~ ~

rd: register destination operand

shamt: shift amount

func: function code (chooses a variant of the operation in op)

- Stored program concept: single memory (store)

for programs and data ("von Neumann" machines)

- Our target machine, TAM, is a two-store machine (a "Harvard" machine)

Different "architecture styles" [Peterson & Hennessy, 1998, p. 201]

- accumulator or "single-register", e.g. IBM 701
- memory-memory, e.g. DEC VAX and IBM 360, which is, however, partly load-store
- load-store: all operations occur in registers
- stack (!!) (TAM is a stack machine)

Advantage: no register allocation problem!