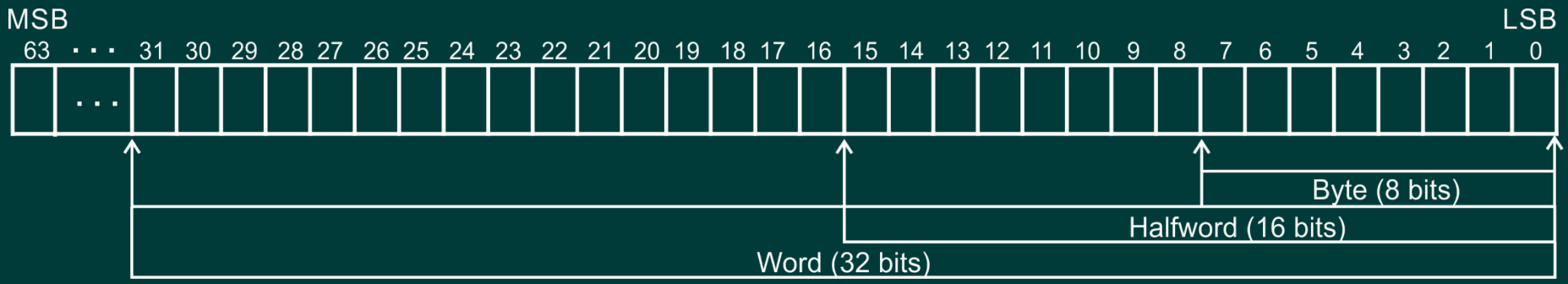
CPU Registers

- Registers are temporary data storage inside the CPU
- The General Purpose Registers (GPR) are used for basic integer operations
  - These are the “architected” registers
  - Architected registers are visible to the assembly language programmer (and compiler)
- There are also special registers that we will identify later
- Registers are one type of operand
  - An operand is a reference to a data item needed by an instruction
  - Depending on the instruction, 1 to 3 operands may be needed

Structure of Registers

- The size of a register in a RISC-V processor is 64 bits



- Bit positions are numbered from 0 to 63
- LSB = least significant bit (bit 0)
- MSB = most significant bit (bit 63)
- Registers may only contain single numeric values:
  - Byte (8 bits)
  - Word (32 bits)
  - Halfword (16 bits)
  - Double Word (64 bits)

RISC-V Registers

- There are 32 general purpose registers (GPR)
- We will not use all of them
  - Some are reserved for special assembler functions
  - Others have special uses that we will not address
- Registers are numbered from 0 to 31
- Alternatively, registers have names for easier use in assembly language programming
  - References to registers in assembly language source programs use the name, not the number
  - The assembler will map the register name to its number in machine code

The Registers We Will Use

- t registers (t0 – t6)
  - Can be used for any data values needed by instructions
  - “t” = temporary (values aren’t not assumed preserved across subroutine calls – more about his later)
- s registers (s0 – s11)
  - Can also be used for any data values
  - “s” = saved (values that must be preserved across subroutine calls – more about this later)
- zero is actually register 0 and always contains the numeric value zero
  - Whenever zero is needed and for copying data values from one register to another; i.e. or t0, t1, zero
- a registers (a0 – a7)
  - “a” = argument (used to pass values to subroutines and for system calls (e.g. end of program)

RISC-V Registers in RARS

- The registers in RARS are displayed in a window on the right of the screen
- There are 32 registers numbered 0 – 31 plus the PC register that is not numbered
- PC cannot be directly accessed from code
- Registers ra, sp, gp and tp will be discussed later

Name	Number	Value
zero	0	0x0000000000000000
ra	1	0x0000000000000000
sp	2	0x00000007fffeffc
gp	3	0x0000000010008000
tp	4	0x0000000000000000
t0	5	0x0000000000000000
t1	6	0x0000000000000000
t2	7	0x0000000000000000
s0	8	0x0000000000000000
s1	9	0x0000000000000000
a0	10	0x0000000000000000
a1	11	0x0000000000000000
a2	12	0x0000000000000000
a3	13	0x0000000000000000
a4	14	0x0000000000000000
a5	15	0x0000000000000000
a6	16	0x0000000000000000
a7	17	0x0000000000000000
s2	18	0x0000000000000000
s3	19	0x0000000000000000
s4	20	0x0000000000000000
s5	21	0x0000000000000000
s6	22	0x0000000000000000
s7	23	0x0000000000000000
s8	24	0x0000000000000000
s9	25	0x0000000000000000
s10	26	0x0000000000000000
s11	27	0x0000000000000000
t3	28	0x0000000000000000
t4	29	0x0000000000000000
t5	30	0x0000000000000000
t6	31	0x0000000000000000
pc		0x0000000000400000

Program Counter Register

- A program resides in memory during execution.
  - the instructions that make up the program are read sequentially from memory by the CPU
- The program counter (PC) is a special register
  - also known as an instruction counter, instruction pointer, instruction address register or sequence control register.
  - part of the datapath
  - not directly accessible through programming
  - initialized by operating system when a program is loaded
  - used to access the instructions in memory during execution

Program Counter Register in RISC-V

- RISC-V instructions are always 4 bytes
  - the program counter increments by 4 after each instruction is read from memory
  - this is automatically done in the hardware
- After an instruction is read from memory, the PC is incremented immediately
  - PC value is then the address of the next instruction to be read from memory
  - While the PC value cannot be directly changed, its value can be altered through programming
    - Branches, subroutine calls and returns