

HBO ICT

hoofdfase

HBO ICT

TI

*Oefenopgaven (assembler)*

**C++ Programming & Software Engineering 1**

**TICT-V1OOPC-16**

Studiejaar 2017-2018

|  |  |
| --- | --- |
|  | Gerelateerde afbeelding |
| Cursuseigenaar | Wouter van Ooijen |
| **Auteur(s)** | Wouter van Ooijen |
| **Datum** | 2017-08-28 |
| Versie | 2.0 |
|  | |

© Institute for ICT, Hogeschool Utrecht, 2017

# Assembler

Dit zijn oefenopgaven voor assembler. De eersten met uitwerking, de rest zonder. De gegeven uitwerkingen zijn zeker niet de enige mogelijke uitwerkingen! Als arrays worden doorlopen, kan je dit proberen met pointers en met array indices.

In alle gevallen is de opdracht de opschreven en/of in C/C++ gegeven functionalitiet te schrijven in (Cortex M0) assember.

## strlen

return de lengte van de string (const char \*) parameter – iteratief

|  |
| --- |
| int strlen( const char \* s ){  int n = 0;  while( \*s != ‘\0’ ){  ++n;  ++s;  }  return n;  } |

|  |
| --- |
| // R0 == s  // R1 == n  // R2 == \*s  strlen:  ldr r1, =0  loop:  ldrb r2,[ r0 ]  add r2, r2, #0  beq done:  add r0, r0, #1  add r1, r1, #1  b loop  done:  mov pc, lr |

Idem, maar nu recursief (best lastig)

|  |
| --- |
| int strlen( const char \* s ){  if( \*s == ‘\0’ ){  return 0;  }  return 1 + strlen( s + 1);  } |

|  |
| --- |
| // R0 == s  // R1 == \*s  strlen:  ldrb r1,[ r0 ]  add r1, r1, #0  bne deeper  ldr r0, =0  mov pc, lr  deeper:  push { lr }  add r0, r0, #1  bl strlen  add r0, r0, #1  pop { lr } |

## vervang

vervang in string s alle ‘old’ characters door ‘new’ – iteratief

|  |
| --- |
| void replace( char \* s, char old, char new ){  while( \*s != ‘\0’ ){  if( \*s == old ){  \*s = new;  }  ++s;  }  } |

|  |
| --- |
| // R0 == s  // R1 == old  // R2 == new  // R3 == \*s  replace:  ldrb r3,[ r0 ]  cmp r3, #0  beq done:  cmp r3, r1  bne dont\_replace  strb r2, [ r0 ]  dont\_replace:  add r0, r0, #1  b replace  done:  mov pc, lr |

## count

return het aantal keren dat x in s voorkomt – iteratief

|  |
| --- |
| int count( char \* s, char c ){  int n = 0;  while( \*s != ‘\0’ ){  if( \*s == c ){  ++n;  }  ++s;  }  return n;  } |

|  |
| --- |
| // R0 == s  // R1 == c  // R2 == n  // R3 == \*s  count:  ldr r2, =0  loop:  ldrb r3,[ r0 ]  cmp r3, #0  beq done:  cmp r3, r1  bne dont\_count  add r2, r2, #1  dont\_replace:  add r0, r0, #1  b replace  done:  mov r0, r2  mov pc, lr |

Idem, nu recursief

|  |
| --- |
| int count( char \* s, char c ){  if( \*s == ‘\0’ ){  return 0;  }  return  (( \*s == c ) ? 1 : 0 )  + count( s + 1, c );  } |

|  |
| --- |
| // R0 == s  // R1 == c  // R2 == \*s  // R5 == 1 or 0  count: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  ldrb r2,[ r0 ]  cmp r2, #0  bne deeper  ldr r0, #0  mov pc, lr  deeper:  push { lr, r5 }    ldr r5, =0  cmp r2, r1  bne no\_match:  ldr r5, =1  no\_match:  add r0, r0, #1  bl count  add r0, r0, r5  pop { r5, lr } |

## Multiply

vemenigvuldig zonder de mul instructie (herhaald optellen)

|  |
| --- |
| int multiply( unsigned int x, unsigned int y ){  int n = 0;  while( y > 0 ){  n += x;  --y;  }  return n;  } |

|  |
| --- |
| // R0 == x  // R1 == y  // R2 == n  multiply:  ldr r2, =0  loop:  cmp r1, #0  beq done:  add r2, r2, r0  sub r1, r1, #1  b loop  done:  mov r0, r2  mov pc, lr |

## Vector add

Tel twee vectoren bij elkaar op, ieder n lang, stop resultaat in een derde vector.

|  |
| --- |
| void add( int z[], int x[], int y[], int n ){  while( --n >= 0 ){  \*z++ = \*x++ + \*y++;  }  } |

|  |
| --- |
| // R0 == z[]  // R1 == x[]  // R2 == y[]  // R3 == n  // R4 == \*x  // R5 == \*y  add:  push { lr, r4, r5 }  loop:  sub r3, r3, #1  blt done  loop:  ldr r4, [ r1 ]  ldr r5, [ r2 ]  add r4, r4, r5  str r4, [ r0 ]  add r0, r0, #4  add r1, r1, #4  add r2, r2, #4  b loop  done:  pop { pc, r4, r5 } |

## Dot-product

return het ‘dot-product’ (innner product) van de twee vectoren. Beide vectoren zijn n elementen lang.

|  |
| --- |
| int dot\_product( int a[], int b[], int n ){  int sum = 0;  for( int i = 0; i < n; i++ ){  sum += a[ i ] \* b[ i ];  }  return sum;  } |

## Spiegel

Spiegel de string: de eerste letter wordt de laatse, etc. Je mag strlen() gebruiken (die hoef je dan niet zelf te schrijven.

|  |
| --- |
| void morrir( char \*s ){  int len = strlen( s );  for( int i = 0; i < len / 2; ++i ){  char temp = s[ i ];  s[ i ] = s[ len – i ];  s[ len – i ] = temp;  }  } |