Jon Bowen

CS 3210

In Class Exercise 2/1/2018

Cover Letter Discussions

1. In this exercise, translating HLL to PAL is easier. Because the higher level language is more expressive, there are fewer forms of a given structure. Test variables must simply be assigned to convenient registers. Then any other variables referenced in the block following the test must be moved to registers. At that point each of the examined test conditions, < and <=, have well defined forms that are acceptable. The statements in the body can then be placed appropriately within that structure.

However, were this not a simplified exercise, there would be additional complications relating to registers. We must assume that the registers we use are, in fact, free to be used. We must also assume that we need not make use of more than 8 registers. PAL does not offer memory access, so nothing can be stored and recalled later. A more complete example of translation would need to account for which registers are in use, perhaps using a stack to track this, and indicate an error if a program required too many registers at one time.

It is also interesting to note that since our only test instructions are BGT and BEQ, the structure of (a < b) tests and (a <= b) tests must be different. We can use BGT b, a, label to take the execution path if (a < b) is true in the first case whereas we can use BGT a, b, label to take the execution path if (a <= b) is false. In one case below we did use BEQ where this same arrangement could have been used, but that was only to limit the number of tests needed to be done, improving execution time.

1. Calls to subroutines in the HLL would be translated by placing parameters in appropriate registers and then unconditionally branching to the label marking the start of the subroutine translation. Upon completion, returned values must be transferred to appropriate registers again, then execution would unconditionally branch to the label indicating the return point of the subroutine.

PAL and HLL Code Fragments

TOP TEST LOOPS

\*\*\* Top Test Loop 1 \*\*\*

//\*\*\*\*\*\*\*\*\*\*\* HLL

int i = 0;

int sum = 0;

while (i < 2){

sum = sum + i;

i = i + 1;

}

//\*\*\*\*\*\*\*\*\*\*\* PAL

; R0: sum

; R1: loop counter

; R2: test value

MOV 0, R1

MOV 0, R0

MOV 2, R2

test:

BGT R2, R1, itr

BR fin

itr:

ADD R0, R1, R0

INC R1

BR test

fin:

END

\*\*\* Top Test Loop 2 \*\*\*

//\*\*\*\*\*\*\*\*\*\*\* HLL

int i = 1;

int prod = 1;

while (i <= 2){

prod = prod \* i;

i = i + 1;

}

//\*\*\*\*\*\*\*\*\*\*\* PAL

; R0: prod

; R1: loop counter

; R2: test value

MOV 1, R1

MOV 1, R0

MOV 2, R2

loop:

BGT R1, R2, fin

MUL R0, R1, R0

INC R1

BR loop

fin:

END

BOTTOM TEST LOOPS

\*\*\* Bottom Test 1 \*\*\*

//\*\*\*\*\*\*\*\*\*\*\* HLL

int i = 0;

int sum = 0;

do {

sum = sum + i;

i = i + 1;

} while (i < 2)

//\*\*\*\*\*\*\*\*\*\*\* PAL

; R0: sum

; R1: loop counter

; R2: test value

MOV 0, R1

MOV 0, R0

MOV 2, R2

loop:

ADD R0, R1, R0

INC R1

BGT R2, R1, loop

END

\*\*\* Bottom Test 2 \*\*\*

//\*\*\*\*\*\*\*\*\*\*\* HLL

int i = 1;

int prod = 1;

do {

prod = prod \* i;

i = i + 1;

} while (i <= 2)

//\*\*\*\*\*\*\*\*\*\*\* PAL

; R0: prod

; R1: loop counter

; R2: test value

MOV 1, R1

MOV 1, R0

MOV 2, R2

loop:

MUL R0, R1, R0

INC R1

BGT R2, R1, loop

BEQ R2, R1, loop

END

IF STATEMENTS

\*\*\* If 1 \*\*\*

//\*\*\*\*\*\*\*\*\*\*\* HLL

int i = 2;

int num = 3;

if (i < 4){

num = num \* num;

}

//\*\*\*\*\*\*\*\*\*\*\* PAL

; R0: num

; R1: loop counter

; R2: test value

MOV 2, R1

MOV 3, R0

MOV 4, R2

BGT R2, R1, doif

BR fin

doif:

MUL R0, R0, R0

fin:

END

\*\*\* If 2 \*\*\*

//\*\*\*\*\*\*\*\*\*\*\* HLL

int i = 2;

int num = 3;

if (i <= 4){

num = num \* num;

}

//\*\*\*\*\*\*\*\*\*\*\* PAL

; R0: num

; R1: loop counter

; R2: test value

MOV 2, R1

MOV 3, R0

MOV 4, R2

BGT R1, R2, skip

MUL R0, R0, R0

skip:

END

IF-ELSE STATEMENTS

\*\*\* If-Else 1 \*\*\*

//\*\*\*\*\*\*\*\*\*\*\* HLL

int i = 2;

int num = 3;

if (i < 4){

num = num \* num;

}else{

num = num \* num \* num;

}

//\*\*\*\*\*\*\*\*\*\*\* PAL

; R0: num

; R1: loop counter

; R2: test value

MOV 2, R1

MOV 3, R0

MOV 4, R2

BGT R2, R1, doif

COPY R0, R7

MUL R0, R0, R0

MUL R0, R7, R0

BR fin

doif:

MUL R0, R0, R0

fin:

END

\*\*\* If-Else 2 \*\*\*

//\*\*\*\*\*\*\*\*\*\*\* HLL

int i = 2;

int num = 3;

if (i <= 4){

num = num \* num;

}else{

num = num \* num \* num;

}

//\*\*\*\*\*\*\*\*\*\*\* PAL

; R0: num

; R1: loop counter

; R2: test value

MOV 2, R1

MOV 3, R0

MOV 4, R2

BGT R1, R2, else

MUL R0, R0, R0

BR fin

else:

COPY R0, R7

MUL R0, R0, R0

MUL R0, R7, R0

fin:

END

Logic for Converting HLL Code Fragments

while (i < 2):

HLL form:

int i = 0;

int sum = 0;

while (i < 2){

sum = sum + i;

i = i + 1;

}

Pseudocode to convert:

MOVE R1, <boolean LHS>

MOVE R2, <boolean RHS>

for (other used variables in loop)

MOVE Rn, var (n = 0, 3, 4,…, 7 as necessary)

test:

BGT R2, R1, itr

BR fin

itr:

<do loop operations>

BR test

fin:

<continue execution>

while (i <= 2):

HLL form:

int i = 1;

int prod = 1;

while (i <= 2){

prod = prod \* i;

i = i + 1;

}

Pseudocode to convert:

MOVE R1, <boolean LHS>

MOVE R2, <boolean RHS>

for (other used variables in loop)

MOVE Rn, var (n = 0, 3, 4,…, 7 as necessary)

loop:

BGT R1, R2, fin

<do loop operations>

BR loop

fin:

<continue execution>

do{…}while (i < 2):

HLL form:

int i = 0;

int sum = 0;

do {

sum = sum + i;

i = i + 1;

} while (i < 2)

Pseudocode to convert:

MOVE R1, <boolean LHS>

MOVE R2, <boolean RHS>

for (other used variables in loop)

MOVE Rn, var (n = 0, 3, 4,…, 7 as necessary)

loop:

<do loop operations>

BGT R2, R1, loop

<continue execution>

do{…}while (i <= 2):

HLL form:

int i = 1;

int prod = 1;

do {

prod = prod \* i;

i = i + 1;

} while (i <= 2)

Pseudocode to convert:

MOVE R1, <boolean LHS>

MOVE R2, <boolean RHS>

for (other used variables in loop)

MOVE Rn, var (n = 0, 3, 4,…, 7 as necessary)

loop:

<do loop operations>

BGT R2, R1, loop

BEQ R2, R1, loop

<continue execution>

if (i < 4):

HLL form:

int i = 1;

int prod = 1;

do {

prod = prod \* i;

i = i + 1;

} while (i <= 2)

Pseudocode to convert:

MOVE R1, <boolean LHS>

MOVE R2, <boolean RHS>

for (other used variables in if block)

MOVE Rn, var (n = 0, 3, 4,…, 7 as necessary)

BGT R2, R1, doif

BR fin

doif:

<execute if block statements>

fin:

<continue execution>

if (i <= 4):

HLL form:

int i = 2;

int num = 3;

if (i <= 4){

num = num \* num;

}

Pseudocode to convert:

MOVE R1, <boolean LHS>

MOVE R2, <boolean RHS>

for (other used variables in if block)

MOVE Rn, var (n = 0, 3, 4,…, 7 as necessary)

BGT R1, R2, skip

<execute if block statements>

skip:

<continue execution>

if (i < 4){…}else{…}:

HLL form:

int i = 2;

int num = 3;

if (i < 4){

num = num \* num;

}else{

num = num \* num \* num;

}

Pseudocode to convert:

MOVE R1, <boolean LHS>

MOVE R2, <boolean RHS>

for (other used variables in if block)

MOVE Rn, var (n = 0, 3, 4,…, 7 as necessary)

BGT R2, R1, doif

<execute else block statements>

BR fin

doif:

<execute if block statements>

fin:

<continue execution>

if (i <= 4){…}else{…}:

HLL form:

int i = 2;

int num = 3;

if (i <= 4){

num = num \* num;

}else{

num = num \* num \* num;

}

Pseudocode to convert:

MOVE R1, <boolean LHS>

MOVE R2, <boolean RHS>

for (other used variables in if block)

MOVE Rn, var (n = 0, 3, 4,…, 7 as necessary)

BGT R1, R2, else

<execute if block statements>

BR fin

else:

<execute else block statements>

fin:

<continue execution>

Code Walkthroughs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Top Test Loop 1 |  |  |  |  |
| PAL |  |  |  |  |
|  | R0 | R1 | R2 | Comment |
| MOV 0, R1 | ? | 0 | ? |  |
| MOV 0, R0 | 0 | 0 | ? |  |
| MOV 2, R2 | 0 | 0 | 2 |  |
| BGT R2, R1, itr | 0 | 0 | 2 | 2 > 0, branch to itr |
| ADD R0, R1, R0 | 0 | 0 | 2 |  |
| INC R1 | 0 | 1 | 2 |  |
| BR test | 0 | 1 | 2 |  |
| BGT R2, R1, itr | 0 | 1 | 2 | 2 > 1, branch to itr |
| ADD R0, R1, R0 | 1 | 1 | 2 |  |
| INC R1 | 1 | 2 | 2 |  |
| BR test | 1 | 2 | 2 |  |
| BGT R2, R1, itr | 1 | 2 | 2 | 2 not > 2, fall through |
| BR fin | 1 | 2 | 2 | branch to fin |
| END | 1 | 2 | 2 | R0 = 0 + 1 as expected |
|  |  |  |  |  |
| Top Test Loop 1 |  |  |  |  |
| HLL |  |  |  |  |
|  | i | sum |  | Comment |
| int i = 0; | 0 | NA |  |  |
| int sum = 0; | 0 | 0 |  |  |
| while (i < 2){ | 0 | 0 |  | 0 < 2, enter loop |
| sum = sum + i; | 0 | 0 |  |  |
| i = i + 1; | 1 | 0 |  |  |
| while (i < 2){ | 1 | 0 |  | 1 < 2, enter loop |
| sum = sum + i; | 1 | 1 |  |  |
| i = i + 1; | 2 | 1 |  |  |
| while (i < 2){ | 2 | 1 |  | 2 not < 2, skip loop |
| } | 2 | 1 |  | sum = 0 + 1 as expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Top Test Loop 2 |  |  |  |  |
| PAL |  |  |  |  |
|  | R0 | R1 | R2 | Comment |
| MOV 1, R1 | ? | 1 | ? |  |
| MOV 1, R0 | 1 | 1 | ? |  |
| MOV 2, R2 | 1 | 1 | 2 |  |
| BGT R1, R2, fin | 1 | 1 | 2 | 1 not > 2, fall through |
| MUL R0, R1, R0 | 1 | 1 | 2 |  |
| INC R1 | 1 | 2 | 2 |  |
| BR loop | 1 | 2 | 2 | branch to loop |
| BGT R1, R2, fin | 1 | 2 | 2 | 2 not > 2, fall through |
| MUL R0, R1, R0 | 2 | 2 | 2 |  |
| INC R1 | 2 | 3 | 2 |  |
| BR loop | 2 | 3 | 2 | branch to loop |
| BGT R1, R2, fin | 2 | 3 | 2 | 3 > 2, branch to fin |
| END | 2 | 3 | 2 | R0 = 1 \* 2 as expected |
|  |  |  |  |  |
| Top Test Loop 2 |  |  |  |  |
| HLL |  |  |  |  |
|  | i | prod |  | Comment |
| int i = 1; | 1 | NA |  |  |
| int prod = 1; | 1 | 1 |  |  |
| while (i <= 2){ | 1 | 1 |  | 0 <= 2, enter loop |
| prod = prod \* i; | 1 | 1 |  |  |
| i = i + 1; | 2 | 1 |  |  |
| while (i <= 2){ | 2 | 1 |  |  |
| prod = prod \* i; | 2 | 2 |  |  |
| i = i + 1; | 3 | 2 |  |  |
| while (i <= 2){ | 3 | 2 |  | 3 not <= 2, skip loop |
| } | 3 | 2 |  | prod = 1 \* 2 as expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bottom Test Loop 1 |  |  |  |  |
| PAL |  |  |  |  |
|  | R0 | R1 | R2 | Comment |
| MOV 0, R1 | ? | 0 | ? |  |
| MOV 0, R0 | 0 | 0 | ? |  |
| MOV 2, R2 | 0 | 0 | 2 |  |
| ADD R0, R1, R0 | 0 | 0 | 2 |  |
| INC R1 | 0 | 1 | 2 |  |
| BGT R2, R1, loop | 0 | 1 | 2 | 2 > 1, branch to loop |
| ADD R0, R1, R0 | 1 | 1 | 2 |  |
| INC R1 | 1 | 2 | 2 |  |
| BGT R2, R1, loop | 1 | 2 | 2 | 2 not > 2, fall through |
| END | 1 | 2 | 2 | R0 = 0 + 1 as expected |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Bottom Test Loop 1 |  |  |  |  |
| HLL |  |  |  |  |
|  | i | sum |  | Comment |
| int i = 0; | 0 | ? |  |  |
| int sum = 0; | 0 | 0 |  |  |
| do { | 0 | 0 |  | enter loop |
| sum = sum + i; | 0 | 0 |  |  |
| i = i + 1; | 1 | 0 |  |  |
| } while (i < 2) | 1 | 0 |  | 1 < 2, enter loop |
| sum = sum + i; | 1 | 1 |  |  |
| i = i + 1; | 2 | 1 |  |  |
| } while (i < 2) | 2 | 1 |  | 2 not < 2, exit loop |
| … | 2 | 1 |  | sum = 0 + 1 as expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bottom Test Loop 2 |  |  |  |  |
| PAL |  |  |  |  |
|  | R0 | R1 | R2 | Comment |
| MOV 1, R1 | ? | 1 | ? |  |
| MOV 1, R0 | 1 | 1 | ? |  |
| MOV 2, R2 | 1 | 1 | 2 |  |
| MUL R0, R1, R0 | 1 | 1 | 2 |  |
| INC R1 | 1 | 2 | 2 |  |
| BGT R2, R1, loop | 1 | 2 | 2 | 2 not > 2, fall through |
| BEQ R2, R1, loop | 1 | 2 | 2 | 2 == 2, branch to loop |
| MUL R0, R1, R0 | 2 | 2 | 2 |  |
| INC R1 | 2 | 3 | 2 |  |
| BGT R2, R1, loop | 2 | 3 | 2 | 2 not > 3, fall through |
| BEQ R2, R1, loop | 2 | 3 | 2 | 3 != 2, fall though |
| END |  |  |  | R0 = 1 \* 2 as expected |
|  |  |  |  |  |
|  |  |  |  |  |
| Bottom Test Loop 2 |  |  |  |  |
| HLL |  |  |  |  |
|  | i | prod |  | Comment |
| int i = 1; | 1 | ? |  |  |
| int prod = 1; | 1 | 1 |  |  |
| do { | 1 | 1 |  | enter loop |
| prod = prod \* i; | 1 | 1 |  |  |
| i = i + 1; | 2 | 1 |  |  |
| } while (i <= 2) | 2 | 1 |  | 2 <= 2, enter loop |
| prod = prod \* i; | 2 | 2 |  |  |
| i = i + 1; | 3 | 2 |  |  |
| } while (i <= 2) | 3 | 2 |  | 3 not <= 2, exit loop |
| … | 3 | 2 |  | prod = 1 \* 2 as expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| If 1 |  |  |  |  |
| PAL |  |  |  |  |
|  | R0 | R1 | R2 | Comment |
| MOV 2, R1 | ? | 2 | ? |  |
| MOV 3, R0 | 3 | 2 | ? |  |
| MOV 4, R2 | 3 | 2 | 4 |  |
| BGT R2, R1, doif | 3 | 2 | 4 | 4 > 2, branch to doif |
| MUL R0, R0, R0 | 9 | 2 | 4 |  |
| END | 9 | 2 | 4 | R0 = 9 as expected |
|  |  |  |  |  |
|  |  |  |  |  |
| If 1 |  |  |  |  |
| HLL |  |  |  |  |
|  | i | num |  | Comment |
| int i = 2; | 2 | NA |  |  |
| int num = 3; | 2 | 3 |  |  |
| if (i < 4){ | 2 | 3 |  | 2 < 4, enter if |
| num = num \* num; | 2 | 9 |  |  |
| } | 2 | 9 |  |  |
| … | 2 | 9 |  | num = 9 as expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| If 2 |  |  |  |  |
| PAL |  |  |  |  |
|  | R0 | R1 | R2 | Comment |
| MOV 2, R1 | ? | 2 | ? |  |
| MOV 3, R0 | 3 | 2 | ? |  |
| MOV 4, R2 | 3 | 2 | 4 |  |
| BGT R1, R2, skip | 3 | 2 | 4 | 2 not > 4, fall through |
| MUL R0, R0, R0 | 9 | 2 | 4 |  |
| END | 9 | 2 | 4 | R0 = 9 as expected |
|  |  |  |  |  |
|  |  |  |  |  |
| If 2 |  |  |  |  |
| HLL |  |  |  |  |
|  | i | num |  | Comment |
| int i = 2; | 2 | NA |  |  |
| int num = 3; | 2 | 3 |  |  |
| if (i <= 4){ | 2 | 3 |  | 2 <= 4, enter if |
| num = num \* num; | 2 | 9 |  |  |
| } | 2 | 9 |  |  |
| … | 2 | 9 |  | num = 9 as expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| If-Else 1 |  |  |  |  |
| PAL |  |  |  |  |
|  | R0 | R1 | R2 | Comment |
| MOV 2, R1 | ? | 2 | ? |  |
| MOV 3, R0 | 3 | 2 | ? |  |
| MOV 4, R2 | 3 | 2 | 4 |  |
| BGT R2, R1, doif | 3 | 2 | 4 | 4 > 2, branch to doif |
| MUL R0, R0, R0 | 9 | 2 | 4 |  |
| END | 9 | 2 | 4 | R0 = 9 as expected |
|  |  |  |  |  |
|  |  |  |  |  |
| If-Else 1 |  |  |  |  |
| HLL |  |  |  |  |
|  | i | num |  | Comment |
| int i = 2; | 2 | NA |  |  |
| int num = 3; | 2 | 3 |  |  |
| if (i < 4){ | 2 | 3 |  | 2 < 4, enter if block |
| num = num \* num; | 2 | 9 |  |  |
| } | 2 | 9 |  |  |
| … | 2 | 9 |  | num = 9 as expected |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| If-Else 2 |  |  |  |  |
| PAL |  |  |  |  |
|  | R0 | R1 | R2 | Comment |
| MOV 2, R1 | ? | 2 | ? |  |
| MOV 3, R0 | 3 | 2 | ? |  |
| MOV 4, R2 | 3 | 2 | 4 |  |
| BGT R1, R2, else | 3 | 2 | 4 | 2 not > 4, fall through |
| MUL R0, R0, R0 | 9 | 2 | 4 |  |
| BR fin | 9 | 2 | 4 | branch fin |
| END | 9 | 2 | 4 | R0 = 9 as expected |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| If-Else 2 |  |  |  |  |
| HLL |  |  |  |  |
|  | i | num |  | Comment |
| int i = 2; | 2 | NA |  |  |
| int num = 3; | 2 | 3 |  |  |
| if (i <= 4){ | 2 | 3 |  | 2 < 4, enter if block |
| num = num \* num; | 2 | 9 |  |  |
| } | 2 | 9 |  |  |
| … | 2 | 9 |  | num = 9 as expected |