Binary Translation for Fun and Profit

Andy Goldstein
OpenVMS Engineering
Andy.Goldstein@hp.com

What is Binary Translation?

- Transformation of an executable or shareable image into an equivalent image for another architecture
- Direct execution of “compiled” code
  - Some code may be interpreted
- Interoperates with native OpenVMS environment
  - System services
  - Executable and shareable images
  - OpenVMS files and other resources

Why Translate?

Yes:

- Source code not available
- Not performance critical
- Low usage / not worth porting effort

No:

- Compute intensive / performance critical
  - 4x – 10x compute time penalty over recompile
- Inner mode execution
- Dynamically generated / modified code

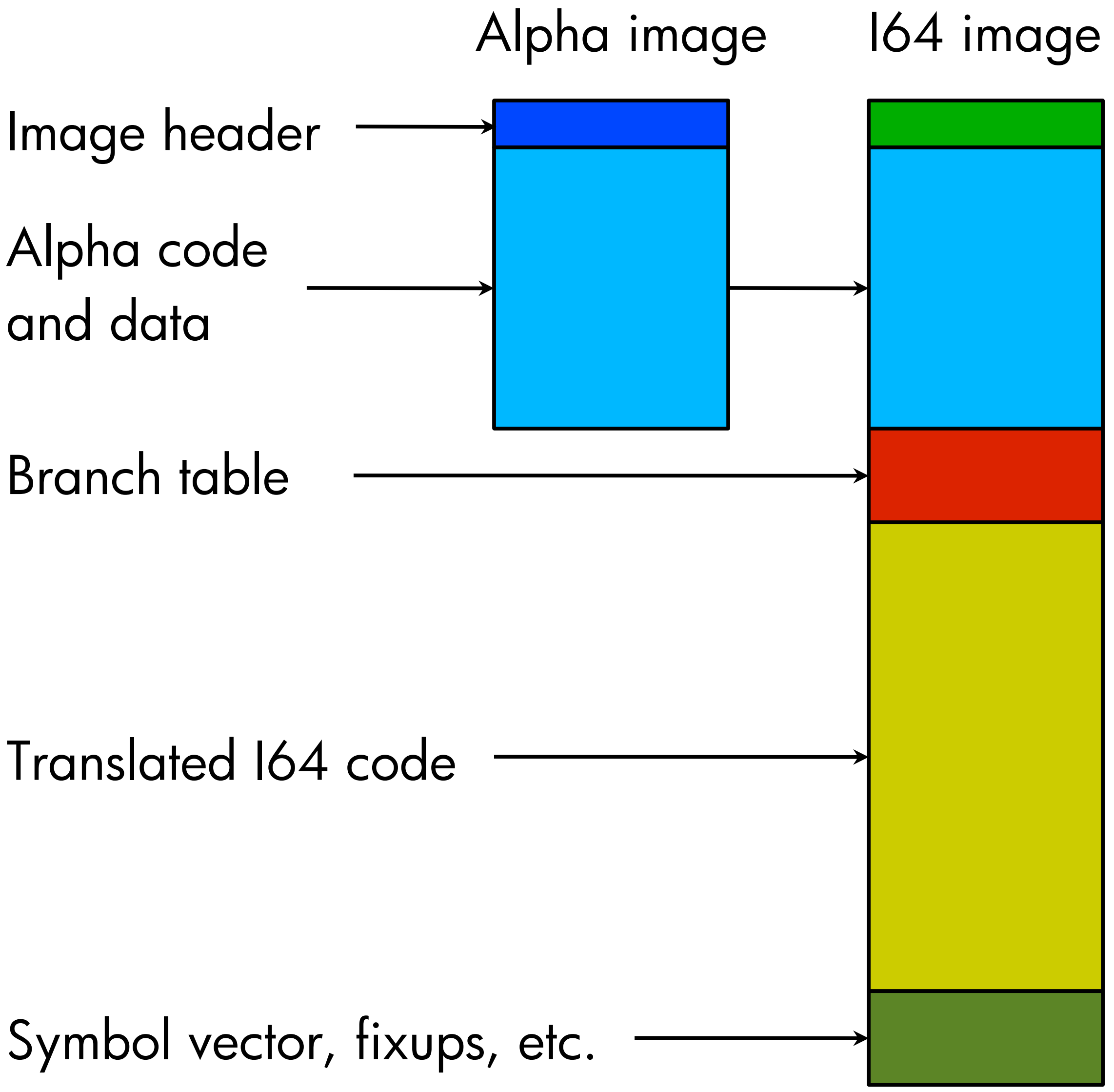
What Can be Translated?

- Main programs and shareable images
- User mode execution only
- Supported languages (RTL support):
  - C, C++
  - Fortran
  - Cobol
  - Bliss
  - Macro
- Programs that work (minimal debug capability)

The Translation Process

- Analyze static branch targets
- Organize source code into basic blocks
- Generate object code for basic blocks
- Build branch table
- Build translated symbol vector, fixup sections, and shareable image list
- Write translated image file

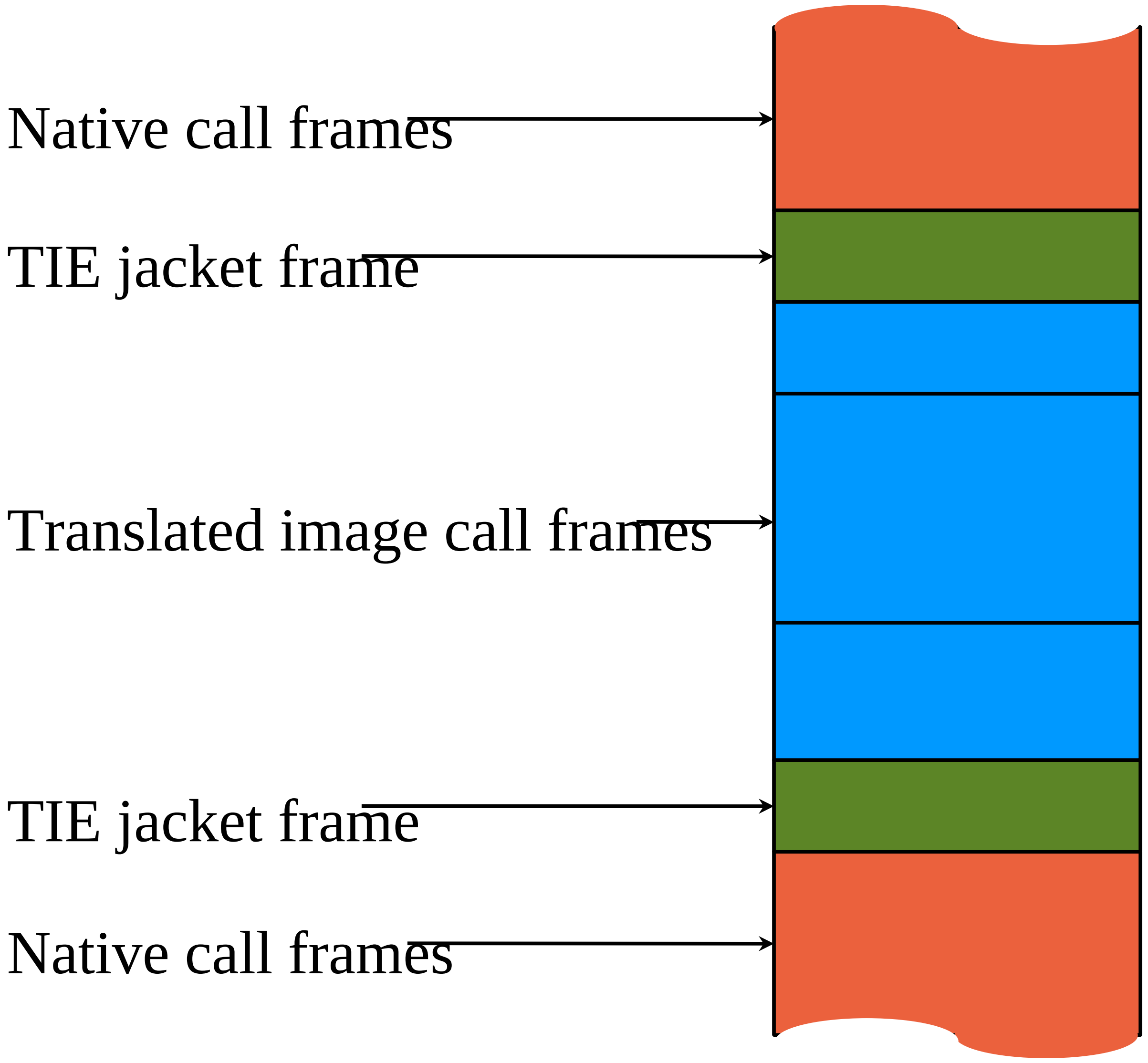
Translated Image Format



Run Time Environment

- TIE = Translated Image Environment
  - Automatically activated for any translated image
  - Call jackets for argument list transformation
  - Translated stack interpretation for exceptions
  - Branch lookup
  - Run time emulation for computed branches / missing code
- Other run time libraries
  - May freely mix and match native & shareable images
  - Must be interface compatible

Translated Image Stack



Shareable Image Compatibility

- Native and translated images may be freely mixed
- Binary compatible interfaces
  - Standard calls
  - Argument order
  - Data formats
- Native images must be compiled /TIE and linked /NONATIVE_ONLY
  - Signatures for argument list translation
  - Outbound calls with OTS$CALL_PROC

Translated Image Names

- Translated image names are suffixed with "_AV"
  - Translated image
  - Shareable image references
- Need logical names for translated / native interoperability
  - e.g., LIBRTL = LIBRTL_AV
  - Included in VMS system startup for all VMS components

Installing

- Free download from http://h71000.www7.hp.com/openvms/products/omsva/omsais.html
  - Enter contact data
  - Execute license agreement
- Download
  - Documentation
  - Translator PCSI kit (Alpha and I64 available)
  - RTL PCSI kit (V8.2 only – integrated with V8.2-1)
- Install PCSI kits

Using the Translator

$ AEST infile

- /EXECUTABLE=outfile (default = infile_AV)
- /AUDIT – just test for translatability
- /INTERPRET – force full interpretation for output image
- /LIST – output summary listing
- /DUMP – output translated image components
- /AIIF=filename – specify image idents and remap symbol vector names

Resource Considerations

- Translated images are large
  - I64 instruction stream encoding
  - Limited optimization opportunities
  - Typically 5x size
- Translator requires large address space
  - May need to increase PGFLQUO

What About the VAX?

- Translation of VAX translated (VESTed) images is supported!
- So…
  - First translate VAX to Alpha with VEST
  - Then translate Alpha to I64 with AEST
- Image name suffixed with "_TV_AV"
- Download VEST from http://h71000.www7.hp.com/openvms/products/omsva/omsva.html
- RTL kit (and V8.2-1 integrated RTLs) include translated VAX RTLs

i n v e n t